DEPARTMENT OF DEFENCE
DEFENCE SCIENCE & TECHNOLOGY ORGANISATION
DSTO

Using THUNDER for Campaign Studies

Peter Maguire

DSTO-TN-0303

Using THUNDER for Campaign Studies

Peter Maguire

Air Operations Division
Aeronautical and Maritime Research Laboratory

DSTO-TN-0303

ABSTRACT

This report provides a detailed guide to the application of the USAF model, THUNDER, to air campaign analysis. All of the steps required to perform an analysis are considered, from setting up and debugging the extensive databases through to running the model and its associated post-processors.

Some of the model's fundamental limitations are also outlined. These limitations place important constraints on the types of campaign analyses which can be conducted with the model. While it is not possible to provide comprehensive solutions to all of these limitations, the flexibility of the model often allows scenario-specific solutions to be implemented. As an illustration, two such problems and their solutions are described in detail.

Published by

DSTO Aeronautical and Maritime Research Laboratory
PO Box 4331
Melbourne Victoria 3001 Australia

Telephone: (03) 9626 7000
Fax: (03) 9626 7999

AR-011-553
August 2000

APPROVED FOR PUBLIC RELEASE

Using THUNDER for Campaign Studies

Executive Summary

This report provides a detailed guide to the use of the THUNDER air campaign model. A brief overview of the model is presented and consideration is given to the basic philosophy behind the model's original construction and thus the types of analyses amenable to THUNDER. The steps required to perform an analysis are described in detail, from setting up and debugging the extensive databases through to running the model and its associated post-processors. Campaign-level models require large amounts of input data from a wide variety of sources. Setting up the input databases for such a model can be a difficult process, and so particular attention is paid to the database construction phase.

Some of the model's fundamental limitations are also outlined. Many of these limitations are a direct result of the model's origins as a USAF air campaign analysis tool during the Cold War. These limitations place important constraints on the types of campaign analyses which can be conducted with the model. While it is not possible to provide comprehensive solutions to all of these limitations, the flexibility of the model often allows scenario-specific solutions to be implemented. As an illustration, two such problems and their solutions are described in detail.

Contents

Nomenclature Lists

THUNDER Executable Programs and Scripts

ttrun	Script which executes a single run of the THUNDER model
ttrep	Script which executes multiple runs of the THUNDER model on a single CPU (serial execution)
ttmultirep	Script which executes multiple runs of the THUNDER model on multiple CPUs (parallel execution)
ttpost	Script which executes the text post processors selected in control.dat
ttgtpp	Script which pre-processes graphical transaction output from THUNDER before graphical post-processors can be executed
ttgraph	THUNDER graph production utility
ttsm	THUNDER graphical utility for replaying the results of a simulation on a two dimensional map
eqkills	Ground equipment kills text post processor
aakills	Air-to-air kills text post processor
sakills	Surface-to-air kills text post processor
muntexp	Air-launched munitions expended text post processor
strtgt	Strategic target attacks text post processor
aaenc	Air-to-air encounters text post processor
rmspp	Reliability, maintainability and supportability text post processor
isrpp	Intelligence, surveillance and reconnaissance text post processor
alftpp	Airlift text post processor

THUNDER Configuration Files

THUNDER.CTL	THUNDER control file, specifying all file locations
ttgtpp.cfg	Configuration file for the graphics transaction post-processor
ttgraph.cfg	Configuration file for the graph generation tool ttgraph
ttsm.cfg	Configuration file for the simulation replay tool ttsm

THUNDER Input Data Files

acmaint.dat	Aircraft maintenance requirements
acplan.dat	Aircraft type specific planning factors
acqcurve.dat	Air-to-ground target acquisition curves
acserv.dat	Aircraft service kit definitions
adaengpb.dat	Ground based air defence engagement probabilities
adclass.dat	Aircraft and air defence class definitions

adcomplx.dat	Air defence complex definition
adsector.dat	Ground air defence sector definitions
advsac.dat	Ground based air defence versus aircraft effectiveness
agwpnmin.dat	Air-to-ground weapon
airairpk.dat	Air-to-air probability of kill values
airalloc.dat	Air mission allocations
airbase.dat	Airbase definitions
airgrdpk.dat	Air-to-ground probability of kill values
airmunt.dat	Air launched munition definitions
airnet.dat	Air movement network definition used for avoiding known defensive concentrations
airplan.dat	Mission specific planning factors
airrules.dat	General air planning rules
aprules.dat	Definition of simple rules affecting air planning
arcs.dat	Surface logistics network arc definitions (road, rail and sea links)
ato.dat	Preplanned air tasking orders
battlfld.dat	Battlefield geometry definition
bpi.dat	Boost phase intercept probabilities versus ballistic missiles
cbg.dat	Carrier battle group definitions
ckpts.dat	Logistics network chokepoint definitions (bridges etc.)
command.dat	Command structure definition
control.dat	Central THUNDER control file
critres.dat	Critical resource definitions
density.dat	Maximum ground force density on the battlefield
detect.dat	
equipsiz.dat	Ground equipment size definitions
grdrules.dat	Ground combat rules
gridcap.dat	Battlefield grid square carrying capacities
harmpk.dat	Anti-radiation missile probability of kill values
intcurve.dat	Interdiction planning factors
intdepth.dat	Interdiction planning factors
intervis.dat	Terrain intervisibility curves definition
isreff.dat	Intelligence, surveillance, reconnaissance system effectiveness data
isrevnts.dat	Intelligence, surveillance, reconnaissance events
isrinit.dat	Intelligence, surveillance, reconnaissance system initialisation information
isrnodes.dat	Intelligence, surveillance, reconnaissance ground nodes
isrplan.dat	Intelligence, surveillance, reconnaissance planning factors
isrsens.dat	Intelligence, surveillance, reconnaissance sensor definitions
isrtgts.dat	Intelligence, surveillance, reconnaissance ground targets
lftevnts.dat	Airlift events
liftato.dat	Airlift preplanned air tasking orders
logfac.dat	Logistics facility definitions
logrules.dat	Ground logistics system rules

mine.dat	Air dropped mine definitions
minepk.dat	Air dropped mine probability of kill values
mobility.dat	Ground unit mobility
muntsurv.dat	Air-to-ground munition survivability curves
nodes.dat	Ground logistics network nodes
ocatgts.dat	Offensive counter air targets
percept.dat	Initial values for perception of the battlefield situation
rngadvn.dat	Air-to-air relative range advantages
saclass.dat	Surface-to-air system class definitions
satellit.dat	Satellite definitions
seedval.dat	Random number seeds
squadron.dat	Squadron definitions
srec.dat	Standoff reconnaissance
stdtgt.dat	Standard target definitions
stitgts.dat	Strategic target interdiction targets
strat.dat	Strategic target definitions
tbmdet.dat	Theatre ballistic missile detectability
tbmunit.dat	Theatre ballistic missile unit definitions
tgtacq.dat	Air-to-ground target acquisition values
tgtavail.dat	Air-to-ground target availability
tolandpk.dat	Takeoff and landing probability of kill values
train.dat	Ground logistics network train/convoy definitions
trnsship.dat	Ground logistics network transshipment point definitions
typeac.dat	Aircraft definitions
typead.dat	Air defence system definitions
typec3.dat	Command, control, communication facility definitions
typeeq.dat	Ground force equipment definitions
typejam.dat	Aircraft jammer definitions
typerdr.dat	Air-to-air and surface-to-air radar definitions
typeunit.dat	Ground unit type definitions
units.dat	Actual ground unit definitions
urgcurve.dat	Resupply urgency curves
weather.dat	Actual and forecast weather definitions
wmd.dat	Weapon of mass destruction definition
wpnvseq.dat	Ground combat probability of kill values
zsprior.dat	Battlefield zone sector resupply priorities

THUNDER Standard Output Files

reports.std	Default report output file, in imperial units
reports.met	Default report output file, in metric units
MISSION.bmn/rmn	BLUE (.bmn) and RED (.rmn) output reports for each specific MISSION
trans.trn	Transaction file, which records all output transactions for further processing
debug.out	Warnings and errors generated during THUNDER execution
game.err	Error file produced when THUNDER crashes due to more serious errors than those reported in debug.out

Military Terms

AAR	Air-to-air refuelling
AEW&C	Airborne early warning & control
ATCAL	Attrition calibration methodology
ATO	Air tasking order
BARCAP	Barrier combat air patrol
C3	Command, control, communications
CAP	Combat air patrol
CAS	Close air support
CBG	Carrier battle group
DCA	Defensive counter air
FLOT	Forward line of offensive troops
GCI	Ground controlled intercept
HVA	High value asset
IADS	Integrated air defence system
INT	Interdiction
ISR	Intelligence, surveillance, reconnaissance
JFACC	Joint force air component commander
JFC	Joint force commander
MOE	Measure of effectiveness
OCA	Offensive counter air
Pk	Probability of kill
RCS	Radar cross section
RECCE	Reconnaissance
SEAD	Suppression of enemy air defences
STI	Strategic target interdiction

1. Introduction

THUNDER is a campaign level warfare model originally developed for the USAF, but now widely used throughout the world. It is capable of simulating air and ground combat, including the air mission planning process, logistics, limited naval operations, surveillance satellites and ballistic missiles. The simulation is fully two-sided, requiring the same level of detail in the input data for both BLUE and RED forces.

Both the air war and elements of logistics are stochastic in nature, meaning that many repetitions of the model are required in order to obtain a measure of statistical confidence in the results. By contrast, the purely ground war elements are deterministic, based on a simple linear piston model. Attrition in the ground war is calculated using the Attrition Calibration (ATCAL) model developed by the U.S. Army Concepts Analysis Agency.

Since THUNDER is such a large model (over 250 000 lines of SIMSCRIPT code), it requires a large amount of input data, both for physical systems and operational doctrine. Very few numbers are hard coded into the model, and it will allow the analyst to try practically anything (it does not understand that air-to-air missiles do not fly at 1 $m.s^{-1}$, for example).

This flexibility is both a blessing and a curse. On the one hand it allows elements which are not normally handled by THUNDER to be modelled approximately, for example some naval warfare elements. On the other hand, the great flexibility means that input data need to be checked carefully for operational validity as there is no "logic checking" by the model itself.

Many aspects of THUNDER can be modelled with varying levels of resolution. This means that parts of the model for which little or no data exists can be represented by rough approximation. Also, the measures of effectiveness (MOE) for some studies may not require all elements to be modelled with high resolution; using low resolution may speed the execution rate and hence reduce post-processing time.

The THUNDER distribution comes with detailed documentation concerning the methodology used in the model and the format of the input and output files. However, it does not go into scenario construction in any detail and so omits a wealth of useful information. Further, operational scenarios relevant to the ADF are totally different in nature to those usually considered by the USAF. As a result, the methods employed by the model are not necessarily capable of fully representing ADF operations.

This document provides a brief introduction to using the THUNDER model for air campaign analyses. Section 2 describes provides a high-level description of the model and its capabilities, which may assist the analyst in deciding if THUNDER is an appropriate methodology for application to particular problems. Sections 3 and 4 detail

the steps required in order to run the model and its associated pre- and post-processors, together with a more detailed examination of the database construction process. Some attention is also paid to the model's limitations in Section 5. A level of familiarity with unix-like operating systems is assumed.

2. The THUNDER Model

THUNDER models the interactions of up to thousands of systems, including aircraft with air-to-air and air-to-surface weapons, air defence sites (including integrated systems), satellites, ballistic missiles, airbases, a variety of ground targets, and inter- and intra- theatre logistics. The operational level command structure is emulated, and can include coalition forces with different Rules-Of-Engagement (ROE).

Being a high-level model, THUNDER does not treat individual systems with high fidelity. Instead, it models the "gross" characteristics of the various entities such that their effects and interrelationships are correctly represented. As such, it relies on accurate effectiveness numbers derived from higher fidelity simulations at the mission, engagement and engineering levels.

The remainder of this section briefly describes some aspects of the model in detail so that the reader may gain an appreciation of its applicability to specific campaign-level analyses. Much of the model's algorithmic detail is contained in the documentation supplied with the model [1]. A discussion of many of the model's limitations is left until Section 5, as this requires a more detailed knowledge of the model's capabilities than is possible before the model is described in detail.

Some important points regarding the model are:

- it models from the Joint Force Commander (JFC)/Joint Force Air Component Commander (JFACC) perspective,
- it includes air, ground and logistics elements,
- it does not explicitly model naval warfare,
- it is fully two-sided, with similar levels of detail required for both sides (BLUE and RED),
- it has little built-in data (only Earth's radius, length of day type information),
- it has a stochastic air war with deterministic ground war, and
- it has variable levels of detail for many air war aspects (including intelligence, surveillance and reconnaisance, weather, air defence calculations); some aspects can have different levels of detail for BLUE and RED.

2.1 Ground War

THUNDER's ground war methodology is based on the US Army Attrition Calibration (ATCAL) model. This is a time-stepped, deterministic model, normally calibrated at the divison level, which considers all combat to occur in one-dimensional pistons. The model was developed during the Cold War era for the analysis of typical NATO versus Warsaw Pact scenarios and may have limited applicability to small scale manoeuvre warfare modelling. The limitations of this method are discussed further in Section 5.

2.2 Air War

THUNDER's air war is stochastic and event-based. The stochastic element relies heavily on a large set of probabilities for virtually all aspects of air-to-air, air-to-ground and ground-to-air combat modelling. For example, the probability that a strike will successfully destroy a ground target, P_{destr}, is dependent on many individual probabilities,

$$P_{destr} = P_{acsurv} \times P_{locate} \times P_{acq} \times P_{discrim} \times P_{launch} \times P_{muntsurv} \times P_k$$

where

P_{acsurv} = probability of platform survival to target area
P_{locate} = probability that platform locates target
P_{acq} = probability that platform/munition pair acquire target
$P_{discrim}$ = probability that platform/munition pair pick the correct aim point
P_{launch} = probability that munition is successfully launched (weapon reliability)
$P_{muntsurv}$ = probability that munition survives the flight to the target
P_k = probability that the munition inflicts the desired level of damage on the target aim point

THUNDER models each of these probabilities seperately. The probability that an aircraft survives to the target depends on many factors, including the properties of the opposing threats and prestrike intelligence, and is modelled explicitly by the aircraft flight path through defended areas. Target location is based on simple calculation taking into account aircraft sensor capabilities and the Intelligence, Surveillance, Reconnaissance (ISR) state (or perception as it is called in THUNDER). The acquisition, discrimination and kill probabilities are modelled explicitly as probabilities for each aircraft, munition and target combination (and possibly day/night and weather dependent as well). Further, the acquisition probability includes all factors which may impede weapon launch, including weapon reliability.

The story is similar for air-to-air engagements. A great deal of information is aggregated into air-to-air P_k values, including such things as evasive manoeuvres and the employment of defensive measures such as chaff and flares. Longer ranged missiles

are modelled through the use of relative range advantages, which describe how many "shots" an aircraft/munition pair can make before all opposing aircraft/munition pairs are within range to return fire.

Although much of the air war methodology appears relatively simple, much detail of the combat is in the form of probabilistic parameters, and careful database construction is required in order to produce realistic results. In some cases, the data may not even exist at the correct levels of aggregation. For example, it may be possible to reproduce exchange ratios obtained using higher fidelity air combat models relatively easily, but getting accurate estimates for engagement probabilities may be more difficult. The exchange ratios are irrelevant if campaign-level estimates of engagement probabilities cannot be obtained.

2.3 Mission Planning and Execution

There are nearly 30 predefined air mission types in THUNDER. These can be roughly broken into several categories:

- air-to-air types such as Combat Air Patrol (CAP) and Defensive Counter Air (DCA),
- air-to-surface types such as Offensive Counter Air (OCA), Close Air Support (CAIRS) and Strategic Target Interdiction (STI),
- support missions such as Suppression of Enemy Air Defences (SEAD), standoff and escort jamming,
- specialist missions incuding Reconnaisance (RECCE), Airborne Early Warning and Control (AEW&C) and Air-to-air refuelling (AAR).

THUNDER generates all of its own Air Tasking Orders (ATOs), taking into account target priorities for each commander as set down by the analyst. A number of factors are fed into the optimisation algorithms so that air resources are efficiently allocated as the analyst requested. Note that the analyst may not have specified an optimum set of allocations. The mission generator takes into account factors such as:

- target values (as specified by the analyst),
- aircraft/munition capabilities,
- target perceptions (intelligence picture), and
- forecast weather (which may be different to the actual weather).

The analyst allocates all air resources to certain mission types, for example strike platforms may be on 50% OCA (versus enemy airbases) and 50% interdiction (versus enemy air defence sites) at the start of a campaign. Occurrences within the campaign can be used to influence air allocation and mission planning through a variety of planning factors, including:

- air resource allocation; e.g., once air defence sites have been sufficiently reduced, some of the strike effort may shift over to OCA or STI versus strategic targets,
- altitude for missions (avoiding certain threats until they have been neutralised),

- weapon selection (choose lower P_k weapon loads with greater strike platform survivability), and
- flight size limitations and the need for escorts, sweeps and Suppression of Enemy Air Defences (SEAD)/jammer support.

The methodology employed is designed to reproduce US Joint Doctrine [2]. There is, however, sufficient flexibility for the methods to be broadly applicable to forces other than the USAF. A good comparison between THUNDER's ATO generator, and that embedded in some other large-scale air models can be found elsewhere [3].

2.4 Logistics Issues

THUNDER is capable of simulating logistics resupply using road, rail, sea and air links. Road, rail and sea links are treated identically during the logistics planning phase, except for drawing transport resources from the stocks of trucks, trains and ships respectively. All resupply issues consider the logistics network as a whole, optimising the route taken given the type of supplies and the urgency of the demand.

Since THUNDER was intended for large-scale operations, individual trucks, trains and ships are not modelled with a high level of detail. All transport assets form a global pool and are available to all logistics facility demands when not being used. It is not possible to specify the initial distribution of ground transport assets.

Air transport is considered differently. Airlift missions are planned in a manner similar for all missions. Calculations for airlift demand are based on analyst input priorities. It is possible to explicitly model the distribution of air assets.

3. Running THUNDER

The THUNDER model is distributed with a number of different executable files, unix shell scripts and simple example databases. The supplied documentation describes these tools in more detail. This section is intended to provide a brief introduction to running THUNDER version 6.5.

Several steps are required to run THUNDER and its associated post-processors. The user needs to:

- set up the THUNDER input files, including setting the desired output in **control.dat,**
- set up the output file structure and create **THUNDER.CTL** (which contains paths of all THUNDER executables, and input and output directories),
- add the Thunder binary directory to user path (TT65/bin),
- run one (**ttrun**) or more (**ttrep, ttmultirep**) repetitions of the THUNDER model, and

- run text and/or graphics post-processors.

Throughout most of this document, it is assumed that a set of input files has been created and that the user is interested in the steps required to run the model and its processing tools. The data set TT65/Data/Datasmall (supplied in the distribution) is used for illustrative purposes. Section 4 discusses many aspects of input database construction.

THUNDER can be run in one of two modes: *analysis* or *game*. In analysis mode, the simulation runs to completion with no user interaction. Multiple repetitions may be executed so that some degree of statistical confidence can be achieved in the results. In game mode, the simulation stops at user-specified intervals, typically at the end of each day. This allows the user to "roll back" the simulation by 1 day to change input data (in certain files) and view the consequences. Throughout most of this document, it is assumed that THUNDER is being executed in analysis mode.

Figure 1 illustrates the sequence of events required in order to run the THUNDER model and all of the associated post-processing tools. Details of input file development are not included here. User input is required in the leftmost box (setting up input) and rightmost box (post-processing of output) while the centre steps (concantenation of input and output) are performed by the model itself. Most of the steps involved in the process are discussed throughout the remainder of this section.

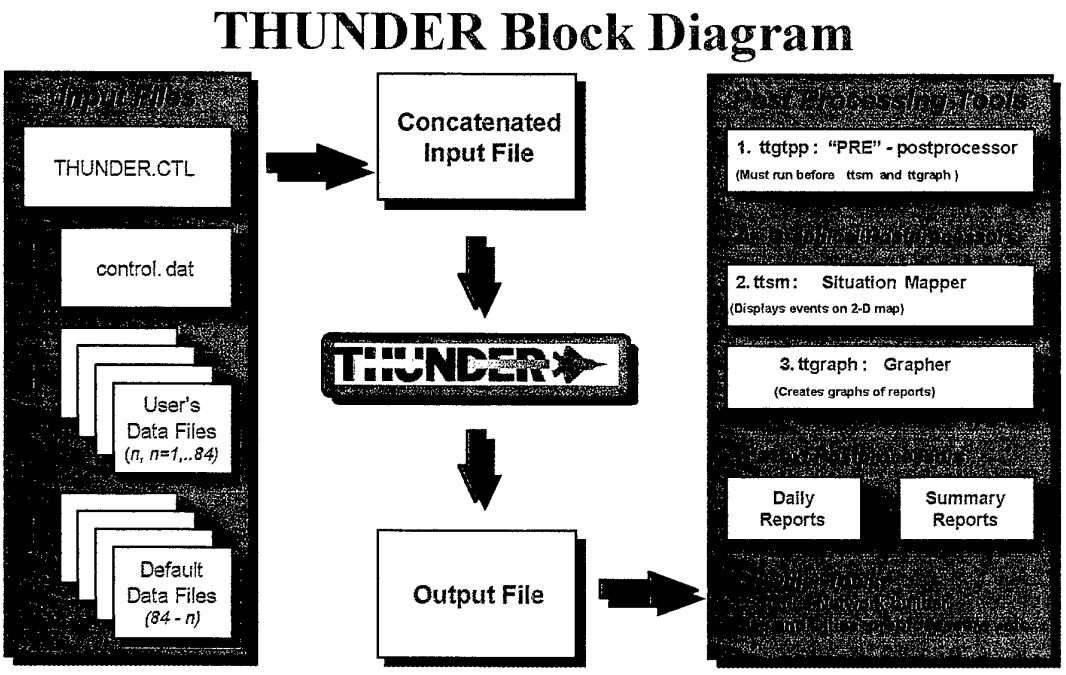

Figure 1: Sequence of events involved in running THUNDER and its associated post-processing tools. Note that in version 6.5 there are 84 user data files.

3.1 Setting Up a New THUNDER Run Area

All output from the THUNDER model and its post-processors is stored in a set of user-specified output directories. In the supplied examples, these are contained in the directory TT65/Run. Each example output directory corresponds to a supplied example input database.

The first step is to set up the base directory for output, e.g., Run/Datasmall. In this directory, create four sub-directories: Reports, Graphics, Backup and Backmods. Reports and Graphics are directories that THUNDER and its post-processors use for output. The last two are only used when THUNDER is running in game mode, but they must be created for the model to run in analyst mode.

The locations of all data input and output directories need to be specified correctly in the file **THUNDER.CTL**. This file must be created in the user's run directory. The file also contains the locations of most THUNDER executables. A sample version of **THUNDER.CTL** is contained in the THUNDER binary directory, and should be copied to the run directory, and edited as required. An example follows. Note that the THUNDER binary directory (TT65/bin) should also be added to the user's path in order to have all programs function correctly. This directory contains not only the actual THUNDER executables, such as **thunder.exe**, but scripts which run the THUNDER model and start post-processing, such as **ttrun**, **ttrep**, and **ttpost**.

The path can either be added permanently to the user's path by appending it to the "set path" instruction in the user's **.cshrc** file (or some other startup file if the c-shell is not being used), or can be added temporarily by typing the following at the command prompt (assuming that the user has the unix c-shell as the default shell):

```
set path=('TT65/bin' $path)
```

where 'TT65/bin' is the full path of the binary directory.

Figure 2 shows an example **THUNDER.CTL**, where the username is joebloggs, and the THUNDER distribution has been installed in the directory /home/THUNDER/TT65.

Several data directories may be specified, but the name of each must contain the string DataDir. When **ttrun** is executed, it searches for data files starting with the current directory and then proceeds upwards from the last directory specified in **THUNDER.CTL**, until a complete set of input files has been found. The command **ttwhich** can be used to check which directory each file is taken from. All four output directories must be specified, but any number of them may be the same directory.

Note also that the full paths for all of the binaries are required in **THUNDER.CTL**, even if the directories have been added to the user's path.

```
# THUNDER.CTL

# This control file contains paths and flags for running THUNDER
# It must be located in the directory you are running THUNDER from

# format: field1 - Variable name (MUST start in first column)
#         field2 - Variable value
#         rest   - comments

# DATA DIRECTORIES (all field names containing "Datadir"):
BslnDataDir                 /home/joebloggs/Base.Data
ModDataDir                  /home/joebloggs/Mod.Data

# OUTPUT DIRECTORIES:
BackmodsDir                 Backmods
BackupDir                   Backup
ReportsDir                  Reports
GraphicsDir                 Graphics

# TTRUN FLAGS:
DoBackups             Y     # (Y/N)
KeepGameTrans         N     # (Y/N)
KeepRegularTrans      Y     # (Y/N)

# EXECUTABLE FILES:
ChecksOffExe          /home/THUNDER/TT65/bin/thunder.exe
ChecksOnExe           /home/THUNDER/TT65/bin/thunderC.exe
TtgraphExe            /home/THUNDER/TT65/Ttgraph/ttgraph.exe
TtgtppExe             /home/THUNDER/TT65/Ttgtpp/ttgtpp.exe
TtsmExe               /home/THUNDER/TT65/Ttsm/ttsm.exe
TtnetExe              /home/THUNDER/TT65/Ttnet/ttnet.exe

# CONTROL DIRECTORIES (for control files used by graphics programs):
TtgraphDir            /home/THUNDER/TT65/Ttgraph
TtsmDir               /home/THUNDER/TT65/Ttsm
TtnetDir              /home/THUNDER/TT65/Ttnet
TtmapDir              /home/THUNDER/TT65/Ttmap

# MACRO DIRECTORY (containing macro data files for Ttgraph program):
MacroDir              /home/THUNDER/TT65/Ttgraph

# TTQUERY CONTROL DIRECTORIES
TtqueryDir            /home/THUNDER/TT65/Ttquery
IncludeDir            DBI64/include
ControlDir            DBI64/control
DbiDir                DBI64
```

Figure 2: Format of the THUNDER.CTL file.

3.2 Generating Output: control.dat

The input file **control.dat**, located with all other input files in the base data directory (e.g., Data/Datasmall), contains control information for the simulation. It can be divided into 12 sections:

- basic simulation controls (number of days, daylight hours etc),
- game controls (only used in game mode),
- air defence calibration controls (not normally used, only useful when calibrating air defence systems),
- computational resolution levels (LOW or HIGH for a number of model areas),
- graphic output controls (including update cycle time for most ground targets),
- overall report controls (standard, metric or both),
- data report controls (data echo reports, useful when debugging databases),
- output report controls,
- game report controls (only used in game mode),
- transaction controls, and
- random number stream data.

In order to obtain specific output from THUNDER, the relevant flags must be set in the file **control.dat**. In most cases, reports can be generated for BLUE, RED or both. The various reports are saved to a number of files in the output directory Reports (see the following section).

Most of the reports are written to a single large file. This file can be produced in either metric units (**reports.met**) or standard (imperial) units (**reports.std**) or both (both files written simultaneously). The version to be produced is specified by the REPORTS.MODE flag in the section OVERALL.REPORT.CONTROLS in **control.dat**. The report file can contain the following:

- data echo reports,
- completed mission reports,
- daily air war mission summaries,
- aircraft maintenance reports,
- ground combat cycle reports,
- air mission planning reports, and
- ground supply cycle reports.

The data echo report flag NUMBER.OF.DATA.REPORTS must be set to the total possible number of reports (84 in the case of version 6.5), regardless of how many are selected for printing (those indicated by YES).

Reports for individual mission kills are stored in separate files for each side (suffixes **bmn** and **rmn** for blue and red respectively, e.g., **barcap.bmn**).

Flags must also be set in **control.dat** (the TRANSACTION.CONTROLS section) for the desired text and graphics post-processors to run (otherwise the desired transactions will not be written). Post-processor flag options are discussed in Section 3.5.

The file **control.dat** is further described in Reference [1], pages 279-302.

3.3 Single Repetition Runs: ttrun

Command syntax:

```
ttrun
```

The **ttrun** script executes a single repetition of the THUNDER model, and is executed from the directory containing **THUNDER.CTL**. No command line parameters are used, but several flags are read from **THUNDER.CTL**. Before running THUNDER itself, the script concatenates all of the data files taken from the one or more data directories, storing the result as a single file called **input.dat**. This file is located in the run directory and is deleted automatically at the end of the run.

All output files produced by **ttrun** are stored in the directory Reports (specified in **THUNDER.CTL**). Below is a list of all output files.

- **reports.std** and/or **reports.met**: standard report file described earlier.
- **trans.trn**: transaction file.
- **debug.out**: warnings generated and errors encountered during execution.
- ***.bmn**, ***.rmn**: mission specific reports for each side.
- **game.err**: error file produced when the model crashes; this occurs when there is a more serious problem than that written to **debug.out.**

Debugging errors in the input databases is discussed in Section 4.10.

3.4 Multiple Repetition Runs: ttrep

Command syntax:

```
ttrep n
```

The **ttrep** script runs multiple repetitions of the THUNDER model in series, each with different random number seeds (thus producing different results). It requires a single command line parameter n (where $n > 1$) giving the number of repetitions to be executed. As with **ttrun**, it should be executed from the base output directory. Note that it cannot be used to run a single repetition.

Many of the files produced by **ttrun** are also produced by **ttrep**, but for each repetition. Transactions are stored for each repetition in the Reports directory (file **trans.trn.REP**, where REP is the number of the repetition), and are compressed. The largest output files, the report files (**reports.std.REP** and/or **reports.met.REP**), are not compressed. It is possible to simply modify the script so that these report files are compressed, but this could increase runtime considerably if all possible reports are being written.

Before executing THUNDER, the **ttrep** script clears any existing output from the Reports and Graphics directories.

Unlike **ttrep** in THUNDER version 6.3, from version 6.4 onwards **ttrep** does not automatically run the graphical post processor **ttgtpp** and the text post processor **ttpost**. In order to automatically run post-processors, user supplied scripts are needed. The locations of these scripts need to be specified in **THUNDER.CTL** by adding the following lines to the end of the file:

```
UserScriptEachRep        TT65/bin/ttuser.each
UserScriptAllRep         TT65/bin/ttuser.all
```

where TT65/bin represents the full path for the scripts. The example above uses the example scripts supplied with THUNDER. Note that there are separate scripts to be executed after each repetition (**ttuser.each**) and at the end of all repetitions (**ttuser.all**). See the following sections for a description of THUNDER text and graphical post-processors. The user is free to write scripts which perform any desired post-processing function. These can be called by **ttrep** is the same manner.

For machines with multiple CPUs there is a version of **ttrep** called **ttmultirep**. This script executes copies of the THUNDER model on each CPU as it becomes available, changing only the random number streams. Apart from running on multiple CPUs, all other functionality is identical to **ttrep**. The number of CPUs is needed as an additional parameter:

```
ttmultirep <number of repetitions> <number of CPUs>
```

3.5 Text Post Processors

Command syntax:

```
ttpost [width] <CR>
```

All of the supplied THUNDER post-processors, whether text or graphics based, perform a similar function: they transform THUNDER-formatted transactions into formats which can be more easily interpreted by the analyst.

THUNDER is supplied with a number of text post-processors. These extract appropriate transactions from the THUNDER output transaction file (or files for multiple repetition runs), generating formatted tables of results, such as air-to-air kills for each side by day. No command line parameters are necessary for a single repetition run using the default page width.

The script **ttpost** runs all of the text post-processors that have flags set in the file **control.dat**. (in the section TRANSACTION.CONTROLS). These flags must be set to a non-zero value (3 for all reports) in order for THUNDER to generate the required transactions in the first place. In the Datasmall example, the post-processor flags are all set to zero and so none are executed by **ttpost**.

Normally the number of repetitions does not need to be specified as a command line option, as the script **ttrep** executes **ttpost** when more than one repetition is performed. It appears that in version 6.5, the number of repetitions is automatically taken into account and the command line option is not necessary. If no output page width is specified, the default value of 80 columns is used (this is also the minimum setting).

The post-processors can produce daily and/or summary reports depending on the value of the corresponding flag in **control.dat** (1=daily, 2=summary, 3=both). These reports are saved in the directory from which **ttpost** is run (i.e. where **THUNDER.CTL** is located).

The six available (analysis transaction) post-processors are:

- **eqkills**: ground equipment kills,
- **aakills**: air-to-air kills,
- **sakills**: surface-to-air kills,
- **muntexp**: air-launched munitions expended,
- **strtgt**: strategic target attacks, and
- **aaenc**: air-to-air encounters.

Other text post-processors provided with THUNDER are:

- **rmspp**: for examining maintenance events,
- **isrpp**: Intelligence, Surveillance and Reconnaissance (ISR) events, and
- **alft**: airlift events.

Text post-processors are discussed fully in Reference [1], Volume 3, pages 49-89.

3.6 Graphical Post Processors

This section is intended to provide a simplified explanation of the use of graphical post-processors. It is not intended to replace the Analyst's Manual section on the subject.

Before discussing individual graphical post-processors, a point needs to be made with regard to configuration files. Most graphical processors (whether pre or post) use a configuration file with the suffix **cfg**, for example, **ttgraph.cfg** for the **ttgraph** post-processor. The locations of many of these are given in **THUNDER.CTL**. However, a local copy of a configuration file (i.e., a copy in the user's run directory) will override any other version of that file.

Also, many processors are executed from scripts located in the TT65/bin directory. Some of the processor executables are located in their own directories, and are called by these scripts, while others are in the TT65/bin directory itself.

Output to be used by the graphical post-processors is produced by setting the relevant flags in the GRAPHIC.OUTPUT.CONTROLS section of **control.dat**. Output can be produced for **ttgraph**, **ttsm** or both.

The program **ttgtpp** is a pre-processor to other graphical post-processors, and after the THUNDER run has completed must be executed before the other graphical processors can be used. This extracts graphical transactions from the file **trans.trn** and puts them into a form useable by the graphical tools. Output is stored in the directory Graphics in the form of several files. Files with the suffix **res** are used by **ttgraph**, while files with the suffix **evt** are used by **ttsm**. The syntax for the ttgtpp command is

```
ttgtpp <configuration file name> <number of replications>
```

and the default configuration file is **ttgtpp.cfg**. If no configuration file is specified, this file must exist in one of the data directories to be searched.

When a multiple iteration run has been performed, **ttgtpp** writes data only for use with **ttgraph** (**ttsm** does not work in multiple repetition mode). All graphics post processors are normally executed from the base output directory.

All of THUNDER's graphical tools are capable of producing output files in encapsulated postscript although it does not exist as a simple menu option. When a graphical tool is launched, a symbol is briefly visible in the top right corner of the window (the letter P in a circle). Although invisible, this symbol can be clicked on at any time, bringing up a dialogue box which allows an image to be saved in postscript. The dialogue box also allows the user to make the postscript symbol visible until the graphical tool is exited (there is no way of permanently setting this; it must be done each time a graphical tool is launched). The location of the symbol can be seen in Figure 3, which shows a **ttgraph** window (which is taken from the Datasmall scenario).

3.6.1 Ttgraph

Command syntax:

```
ttgraph [config file] [-r<macro file>] [-o<report file>]
```

Ttgraph is THUNDER's plotting package. It allows the user to plot on 2D axes a variety of measures of merit, ranging from such things as Forward Line of Offensive Troops (FLOT) movement as a function of time to air losses for each mission type.

Ttgraph can be run in two modes: direct and batch. In direct mode, graphs can be viewed and modified. In batch mode, text reports are produced and no graphical output is generated. Batch mode is used by specifying a macro file on the command line. The default report file is **ttgraph.rpt**, which is saved in the base output directory. Ttgraph should normally be executed from this directory.

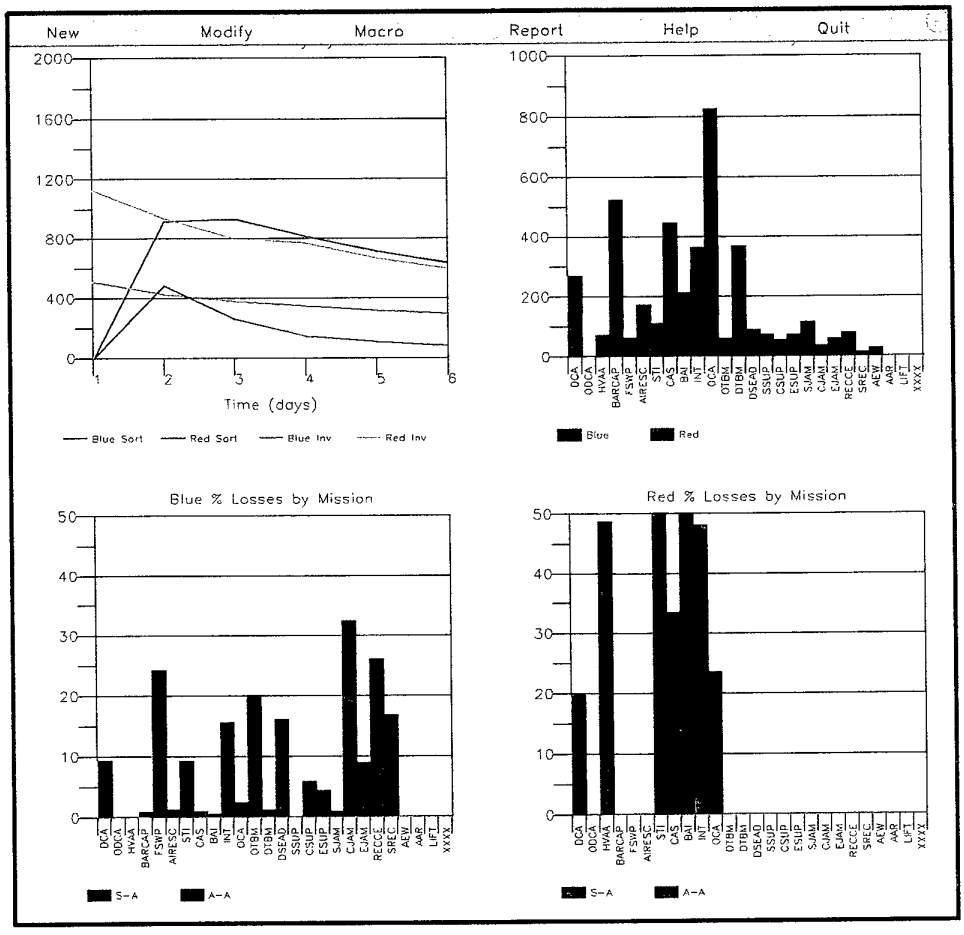

Figure 3: ttgraph window showing the location of the "hidden" print button in the upper righthand corner.

The default configuration file is **ttgraph.cfg**, located in the data directory. This file specifies the location of output files to be used, as well as such things as plot colours and fill styles. A local copy of the file will be used instead if it exists.

The program **ttgraph** allows up to four graphs to be view at once. Each of these can then be modified independently. Data from the graphs can be saved in a text file or can be printed directly. It is possible to save a **ttgraph** format in the form of a macro file. This file can then be read in after subsequent THUNDER runs in order to analyse a number of data sets in a consistent manner. In order to do this, ths user displays the desired set of graphs, and under the Macro menu, selects save.

It is also possible to compare the output from two different scenarios using **ttgraph**. The locations of the graphical output files need to be specified in the configuration file **ttgraph.cfg**. When creating new plots for comparison, **ttgraph** will prompt the analyst for the scenario name, and the process may be repeated to create similar plots for both scenarios. The two plots will be displayed next to each other by default, allowing a quick visual comparison. Note that there is no measure of any error for multiple repetition runs, and so apparent differences may not be statistically significant. None of the supplied THUNDER post-processing tools allow an estimate of errors.

Ttgraph is discussed fully in Reference [1], Volume 3, pages 46-49, including the format for the configuration file.

3.6.2 Ttsm

Command syntax:

```
ttsm [config file]
```

Ttsm is the THUNDER Situation Mapper. It allows the user to graphically view how the simulation proceeded, and can only be used for single repetition runs. As such, it is more useful for debugging and producing presentations than for doing real analysis work. For multiple repetition runs, it is possible to run a separate program called **ttave**, which selects a repetition close to the "average" (which is, to some extent, user definable). This repetition can then be used to produce graphics transactions for use with **ttsm**. See the file README.TXT located in the THUNDER Utils directory for a description of how **ttave** selects the most representative repetition.

The default configuration file is **ttsm.cfg**, located (as are most ***.cfg** files) in the data directory. This file allows the user to set which types of ground objects are displayed on startup, although any other objects can be switched on from the menus once ttsm is started. The user can also specify the default size of the map, so that zooming in or out is not necessary once **ttsm** is running.

Ttsm has a single step animation feature which allows the user to step through the simulation (the step size is specified in **control.dat**, and is 6 hours for Datasmall). After each step, the status of any ground target can be examined (by clicking on Map then Status). For example, an airbase can be viewed to see which squadrons are stationed there (and exactly how many aircraft are currently on the ground), what supplies are on hand, what damage has been sustained etc.

Before beginning the animation, it is possible to switch on flight paths for any number of mission or aircraft types. With many missions switched on, stepping can become quite slow. Once switched on, there is no speed improvement if they are later switched off- the user must exit **ttsm** first.

Below are some figures showing examples from **ttsm**. These are all taken from the Datasmall scenario. Figure 4 shows an overview of the scenario, a large scale NATO versus Warsaw Pact exercise.

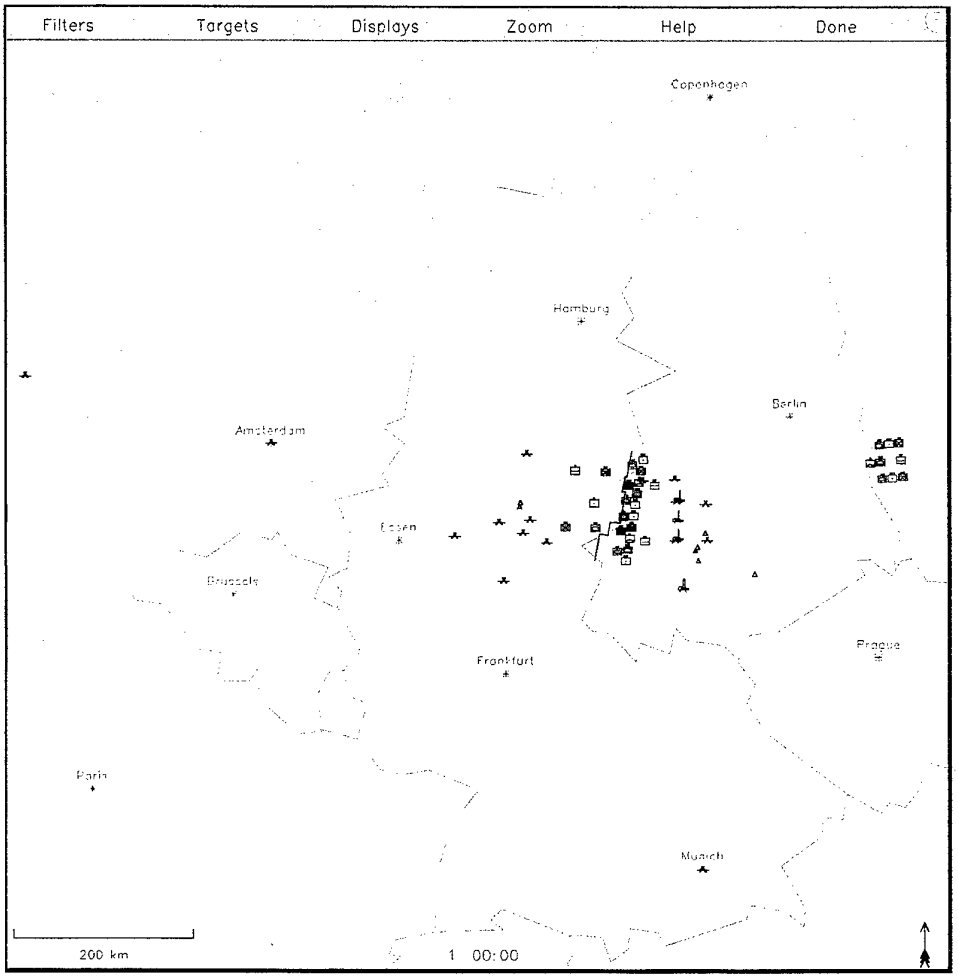

Figure 4: Initial ttsm window showing the Datasmall scenario.

In Figure 5, BLUE offensive counter air (OCA) air missions have been turned on so that flight paths, ground attacks and aircraft attrition can be witnessed. The simulation results were then stepped through until some activity was noticed (first OCA missions began to fly). The dashed lines originating from the airbases indicate the flight paths..

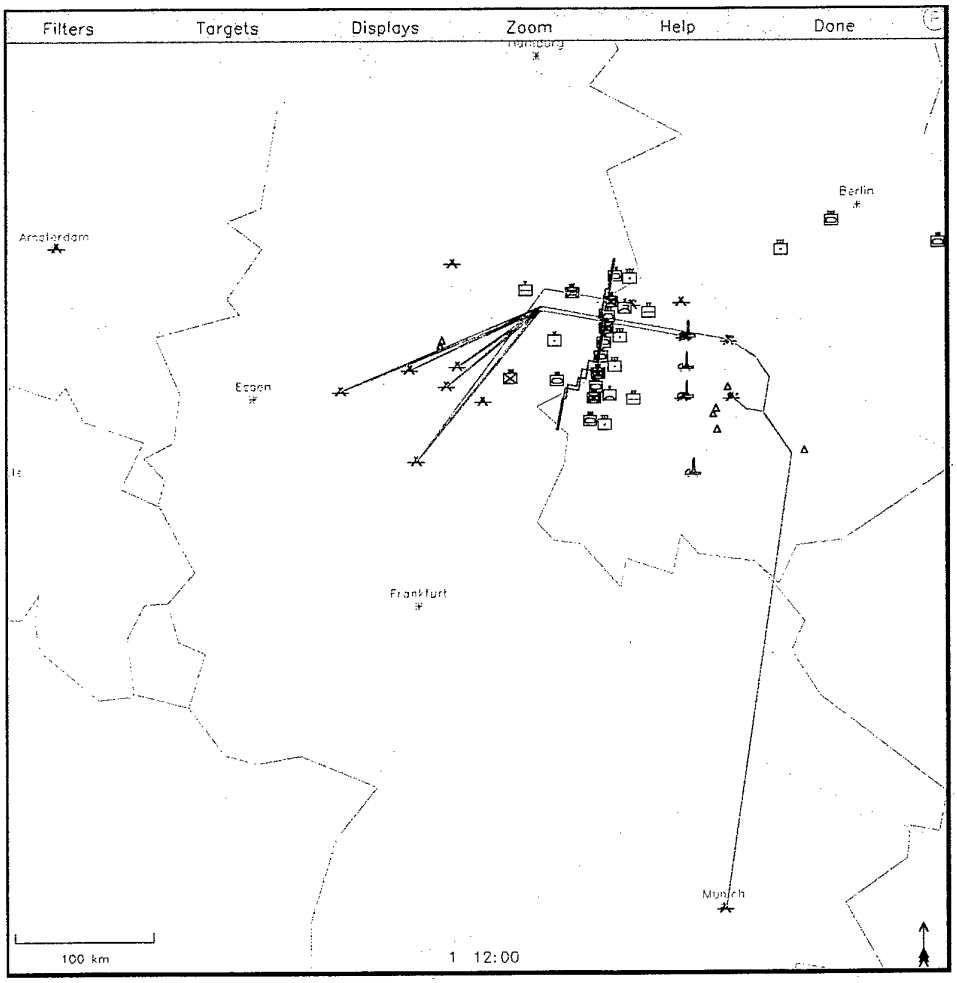

Figure 5: Zoomed ttsm window showing BLUE OCA missions attacking RED airbases

The situation mapper is discussed in Reference [1] Volume 3, pages 38-44.

3.7 Utilities Not Supplied with the THUNDER Distribution

The THUNDER home page (www.s3i.com/Default.htm) contains a link to a set of text based utilities which can be useful during database construction and debugging. Binaries for these utilities can be found in the utilities directory (on the same level as TT64), while the source code is contained in the subdirectory Source. The utilities are:

- **acclosure:** lists the number of each type of aircraft by squadron and airbase.
- **acconfig:** terminal delivery profile data for each platform/load in typeac.dat.
- **adcount:** lists air defence sites, including their input file location.
- **unclosure:** lists various unit properties, such as ID, name, status etc.
- **unitwpns:** lists the equipment in each type of unit, and the sum for the whole scenario.

All of these utilities need to be executed from the normal runtime directory, i.e.,the directory which contains **THUNDER.CTL**. They require no configuration files or command line options. Currently, these utilities are only in the form of SUN binaries and SIMSCRIPT source, and only for version 6.4.

Some other utilities have been written within AOD. These include a set of text-based post-processors which calculate error estimates as well as mean values, and a set of utilities for generating THUNDER format terrain files [4].

4. Input Database Generation

In THUNDER version 6.5, over 80 input files are needed to set up a working scenario. All of these files are in ASCII format and most can be easily read and interpreted. Since there is such a large amount of input required, building a scenario is a complex process which needs to be carried out in a logical manner.

This section supplies some information which should be taken into account when generating new databases or modifying old ones. It is not intended to provide all of the information necessary, but should guide the analyst in setting up a scenario of modest complexity.

A good description of the sequence needed when generating a database is given in Reference [1], Volume 3, pages 20-28. This divides database construction into the following sequence of steps:

- environment,
- ground war,
- air war,

- integrated air defence system (IADS),
- intelligence, surveillance, reconnaissance (ISR),
- ground war planning, and
- air war planning.

Some files need to be modified in more than one location, e.g., **typerdr.dat** which contains both ground based air defence radars and airborne radars, and thus is modified in both the air war and IADS steps. Even though the input files can be notionally divided into these categories, there is still a high degree of cross referencing which the analyst needs to be aware of.

Another possible division of the input data into functional areas is shown in Figure 6. This format breaks the database into a series of progressive steps. For example, it is necessary to construct a list of standard target elements (part 2) before any fixed targets can be defined (part 5). Without fixed targets, air-to-ground probability-of-kill (P_k) values cannot be defined in part 7, and so on. This division of the data is also useful as multiple analysts can effectively work independently on different parts which are at the same level, such as ground objects and air objects. This method is the one used throughout the rest of the document.

Note that areas identified as advanced topics may be omitted on a first attempt at database construction; they are not essential in the construction of a simple scenario. These advanced topics are not discussed in this report.

1. Battlefield Construction	**2. Standard Targets**	
battlfld	Stdtgt	
city: use ttcitypp		
map: use ttmappp		
3. Terrain and Weather	**4. Rear Area Network**	**5. Fixed Targets**
mobility	(use ttnet with blank files)	Strat
density	arcs	trnsship (linked to arcs)
gridcap	nodes	logfac
intervis (and intcurve)	ckpts	airbase
weather		
6. Ground Objects	**7. Air Objects**	**8. Air defence objects**
command	command	Typead
typeeq	squadron	typerdr (also in air objects)
wpnvseq	typerdr (also in ad objects)	adclass
typeunit	typejam	advsac
units	airmunt	muntsurv
equipsiz	typeac	saclass
train	acmaint, acserv	adcomplx
typec3	tgtacq	
(units,cmds,logfacs)	acqcurve	
	agwpnmin	
	mgadvn	
	airairpk,airgrdpk,harmpk	
	mine,minepk	
9. Air/Ground Planning		
airplan	intdepth,intcurve	logrules
acplan	ocatgts	percept
airrules	stitgts	srec
aprules	tolandpk	urgcurve
ato	zsprior	critres
liftato	airnet	detect
airalloc	grdrules	splycrvs
Advanced Topics		
IADS	TBM/WMD	ISR
adsector	bpi	Isreff
adaengpb	tbmdet	Isrevnts
	tbmunit	Isrinit
		Isrplan
	wmd	Isrtgts
		Isrnodes
		Satellit

Figure 6: Decomposition of the THUNDER input files into sequential groups.

4.1 Battlefield Construction

4.1.1 The Battlefield: battlfld.dat

This file specifies the location and orientation of the battlefield on the input map, together with the battlefield size (including the battlefield grid size), FLOT location and the command zones and sectors which make up the battlefield. It contains further information used by a variety of graphical pre and post-processors. For a discussion of battlefield zones and sectors see Reference [1], Volume 1, pages 5-7.

One important note regarding the battlefield origin needs to be made. If the origin coordinates are specified to be negative, both the degrees and minutes part of the origin definition must be made negative.

Once the battlefield size has been specified here, it will not change throughout a simulation. Terrain is only modelled within the battlefield and so it should be large enough so that all important terrain features are included. It is thus important to make the battlefield large enough to cover all of the region of interest. Further, many targets will not be automatically attacked in the air war if they are not on the battlefield.

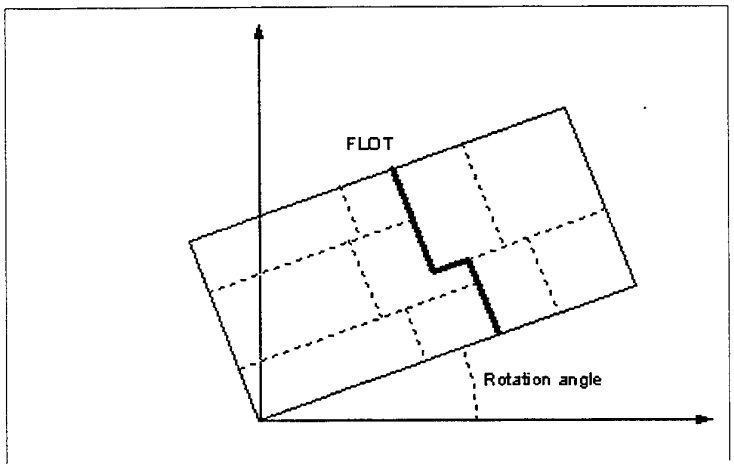

Figure 7: Battlefield geometry, showing a positive rotation angle, the FLOT, and the division of the battlefield into command zones (parallel to the FLOT) and sectors (perpendicular).

The way in which ground force combat is modelled in THUNDER requires care to be taken when setting up the FLOT. Refer to Figure 7. Since ground forces battle across sector segments, disjointed FLOTs can result in a large "no mans land" where no units (or airbases) can operate. This area of territory is essentially dead until the FLOT moves. Dummy units (i.e. units which contain no equipment) can be used to set the FLOT geometry. Since the units have no equipment, they cannot actually take place in

ground combat. Since they are not actually involved in the war, they may have any useful sector width (defined by their superior dummy command) and so just serve to eliminate any large "no mans land" areas.

A number of files contain data which are defined on the battlefield grid (not the zone/sector grid), and so the size of the grid is important when setting these up. These files include:

- **mobility.dat**: Ground unit mobility,
- **intervis.dat**: Intervisibility within the battlefield (terrain masks ground targets etc),
- **density.dat**: Maximum density of ground forces,
- **gridcap.dat**: Grid carrying capacity.

Since the grid size determines the terrain resolution, it should be small enough to describe adequately the actual terrain, but not so small as to require a prohibitively large number of data points. Further discussion of these files can be found in Section 4.3.

Typically, grid square sizes are of the order of 20km to 50km. This could be a problem for large battlefields, as a large number of squares could increase runtime considerably (especially if the same grid is used by the air network). Ignoring the air network, most of the runtime influence is associated with the ground war. The only exception is the intervisibility curves, which are used to calculate lookup angles for air defence sites.

A point needs to be made regarding battlefield zones and sectors. The zone/sector arrangement described in the analyst manual [1] is different to that actually used when setting up the command structure in **battlfld.dat** and **command.dat**. Instead of zones being numbered from the left of the battlefield (as described in the manual) they are actually numbered outwards from the FLOT for each side, so there are two zones numbered 1 etc. THUNDER knows that a RED command or unit located in zone 1 will be on the righthand side of the FLOT.

In version 6.4, zone/sectors also have an effect on surface-to-air engagements. See Section 4.8 for a further explanation. This has been changed in version 6.5.

From THUNDER version 6.4 onwards, the file **battlfld.dat** also contains a description of the air mission planning network. Air missions can be told to fly using the network, avoiding regions with dangerous air defences and analyst defined "no-fly-zones". This network must be at least as large as the battlefield itself, and may have the same origin or a user specified one. It is convenient to set it to the same resolution as the battlefield, although care should be taken to make sure that there are not too many air network grid squares. This is because a form of shortest path algorithm is used to plan missions flown on the network: too many squares can dramatically increase run-time. See the Reference [1], Volume 2, pages 7-9, for a description of the air network.

The air network can also have an effect on air defence engagements. With the appropriate planning factors, it can be used to plan air missions so that flight groups avoid known regions with significant air defence assets. A similar effect can be achieved by simply modifying the air defence engagement probabilities for selected classes of air defence sites (in **adaengpb.dat**) without using the air network.

4.1.2 Generating Maps and Cities: ttmappp and ttcitypp

These two pre-processors take existing ASCII data files in lat/long format and convert them into formats used by THUNDER. There is no on-screen output; they simply take input files and do the conversion internally. Both use simple configuration files, **ttmappp.cfg** and **ttcitypp.cfg** respectively, which specify the input and output filenames. Local versions of these files will override those in the data directory hierarchy, as with data files. These pre-processors are executed directly, ie, no scripts are used. If the battlefield origin and/or rotation is changed in **battlfld.dat**, then both of these utilities will need to be executed. This will ensure that all THUNDER input and output is referring to the same battlefield. If this is not done then problems in the form of apparently inconsistent screen coordinates will occur when attempting to use **ttnet** and **ttsm**.

Each of the post-processors is contained in its own directory within TT65. As such, they will not normally be included in the user's path.

The map preprocessor is invoked using

```
ttmappp <configuration file name>
```

The map pre-processor **ttmappp** takes a file containing lat/long polygon data for the various land masses (see the directory TT65/Ttmappp for some examples; these are files with the extension **.map**) and converts this into a binary file used by THUNDER called **map.dat**. This is the only binary file used by THUNDER, and is only important to the various graphical tools which use the map, such as **ttnet** and **ttsm**. The input file, as stated, is simply a set of polygons. The polygon entries have a very simple form, and it is easy to add new ones, or delete existing ones. An example of the polygon format is given below (a simple rectangle).

```
 2  4  "Square"
 12.5   133.2   12.5   136.7   14.8   136.7   14.8   133.2
```

The first two numbers are the polygon colour (see Ttmappp/colors.cfg) and the number of points. These are followed by an optional polygon name, and on the next line a simple list of the lat/long pairs which make up the polygon. Ttmappp appears to be particular about the format of this input file; there cannot be any control characters anywhere and there must be five points on each line (except for the final line in a polygon).

One other input file required by **ttmappp** is **battlfld.dat**, which contains battlefield orientation information. The ttmappp binary resides in the directory TT65/Ttmappp and can be executed from the data directory containing the lat/long data map. A sample configuration file (**ttmappp.cfg**) can be found in the ttmappp directory.

Accompanying **ttmappp** in the same directory is a related tool called **viewmaps**, which displays the available maps. The standard THUNDER distribution comes supplied with a set of maps covering most parts of the world, including a map of Australia and the immediate region (**australia.map**). Viewmaps calls another utility called **ttmapdraw**, which does the actual drawing. This utility can be used on its own to display a single map. The format is

```
ttmapdraw <centre lat> <centre long> <size in metres> mapname
```

For example, to display the map of Australia, centred on 23°S, 133°E, enter:

```
> ttmapdraw -23 133 7000000 australia.map
```

The city preprocessor can be invoked using

```
ttcitypp <configuration file name>
```

The city preprocessor **ttcitypp** takes a list of cities and converts their lat/long data into screen coordinates for use with tools such as **ttsm**. The city data is not used by THUNDER in any way other than for reference points on graphical output; cities cannot be targeted and contain no combat assets. Ttcitypp, as with ttmappp, does not allow control characters to be embedded in the input data.

As mentioned earlier, both of these pre-processors make use of the file **battlfld.dat**, and so both must be re-run (and thus new **map.dat** and **city.dat** files produced) whenever **battlfld.dat** is modified. This is necessary regardless of how trivial the change is, otherwise the graphical post-processors will not show a true location of all objects.

4.2 Standard Targets

All targets which can be attacked from the air in the THUNDER model are composed of standard target instances. Each of these standard targets consists of a number of elements and has a bounding radius (e.g., three tanks within 100 metres). Each standard target also has a target acquisition and discrimination probability, which can be unique. These probabilities, together with the air-to-ground probability-of-kill (P_k) can depend on aircraft/munition pair and weather band.

Since all air-to-ground combat is dependent on the standard target set, care should be taken during its construction. Also, for this reason, the subject is discussed in some detail.

When setting up the list of standard targets, it is a good idea to insert duplicates of many elements, such as shelter 1, shelter 2 etc, with distinct graphic IDs. This is done for two reasons. First, it reduces the need to edit the standard target file at a later date (which could cause ID conflicts). Second, it allows otherwise identical target instances allocated to different targets to be identified.

Consider an example where there is only one type of shelter allocated to two RED airbases. If BLUE attacks both airbases and destroys some shelters, all kills will be grouped together in **ttgraph**; four shelters killed. It is possible to search through all of the transactions and find out how many shelters were killed at each airbase, but there is a much simpler solution. If each airbase has its own type of shelter target (they may be identical) then ttgraph will simply show that, for example, three shelters of type 1 (at airbase 1) and one shelter of type 2 (at airbase 2) were destroyed.

The set of standard targets is contained in the file **stdtgt.dat**. This must be constructed before any other targets, such as ground units, airbases, and air defence sites, can be defined. For each standard target defined in this file, the following data are needed:

- name and ID (set by user; all files with targets refer to these),
- number of elements (e.g., 3 tanks, single fuel truck, one hardened aircraft shelter),
- radius covering all elements,
- acquisition class (refers to **tgtacq.dat**),
- discrimination class (refers to **tgtacq.dat**),
- graphic ID (used by graphical tools such as **ttsm** and **ttnet**), and
- ISR target perception ID.

In all other files which contain ground targets, the objects are then composed of combinations of standard target instances. For example, an airbase may have two runway standard targets, four POL storage tanks, six hardened aircraft shelters and so on. Probability-of-kill data for each aircraft munition pair versus each standard target, by weather band (**weather.dat**) is contained in the file **airgrdpk.dat**. Below is a list of all files which contain standard target references:

- airbases (**airbase.dat**),
- ground units (**units.dat**),
- choke points (**ckpts.dat**),
- transshipment points (**trnsship.dat**),
- air defence site type definitions (**typead.dat**), and
- C3 facility type definitions (**typec3.dat**).

If two targets are located at the same point (or are very close), and one of the targets is attacked, no collateral damage effects are considered as THUNDER treats each target seperately. In order for collateral damage to be modelled, the individual targets must

be separate target instances within a single large target (such as an airbase or strategic target).

The USAF constructs THUNDER targets using a combination of Joint Munitions Effectiveness Manual (JMEM) data to define ground targets and simple P_ks, and output from the Saber Selector (SABSEL) model for the more "complex" P_k values.

4.3 Terrain and Weather

THUNDER does not model terrain explicitly. Instead, it models the effects of terrain on such things as ground movement and air defence engagements. These effects are contained in the following set of files:

- **mobility.dat**: ground unit mobility.
- **intervis.dat**: intervisibility within the battlefield (terrain masks ground targets etc); this is used in conjunction with **intcurve.dat**, which contains the actual intervisibility curves.
- **density.dat**: maximum density of ground forces.
- **gridcap.dat**: grid carrying capacity.

Thus each battlefield grid square has allocated to it a total of four numbers which model the terrain. Note that the intervisibility curves only affect surface-to-air engagements. No terrain effects are taken into account off the battlefield, although the curvature of the earth is still used in air-defence calculations.

Also, THUNDER cannot discriminate between water and land. It is up to the analyst to set appropriate values for the above numbers for any grid squares which encompass bodies of water.

Entering all four numbers into the matrices can be a tedious and error prone experience, especially if there are a large number of grid squares (which may be the case for a large battlefield). Tools developed within AOD can be used to automate the process, using digital elevation maps of the region of interest. It remains up to the analyst to decide on a set of terrain types, and what the four numbers corresponding to each terrain type should be, but once this is done the creation of the four matrices is automatic. These tools will be described in a future report [4].

The weather to be used during the simulation is contained in the file **weather.dat**. This is divided into four sections. The first is a set of weather bands. Each band has a ceiling in feet and a visibility in nautical miles. These numbers are used to construct all weather. The second section contains the locations of a number of weather stations. THUNDER does not keep track of the weather at every point on the map, only at these stations; the weather at all other points is interpolated or extrapolated from these values. The third section divides the day into a set of day segments. The weather can only change from one day segment to the next.

Finally, the weather file contains a table of the weather band for each day, day segment, and station. If the number of days of weather is less than the number of days of simulation, then the last day of weather is simply repeated. This table needs to be repeated for both actual and forecast weather, which may be different. All mission planning is done using the forecast weather.

Parts of the THUNDER model affected by weather include:

- mission planning: only aircraft/weapons pairs which can be used in the forecast weather over the target are available,
- airbase operations: each airbase has a minimum ceiling and visibility for operations to continue,
- target location/acquisition/discrimination: actual weather might not enable the platforms to deliver their munitions, and
- air defence engagements: air defence site types can have their capabilities reduced by poor weather and/or day/night.

When setting up a scenario initially, it is a good idea to start out with perfect weather (high ceilings and long range visibility everywhere on all days). The consequence of this is that no mission will be cancelled or targets not located due to bad weather. Hence, if missions are not flying, then it must be due to some other factor.

4.4 The Logistics Network

THUNDER can be run either with or without a logistics network. If there is no network, units and airbases will not be resupplied and combat will grind to a halt when shortages occur. (Initially, squadrons deploy with one service kit, identified in **squadron.dat**.) Having no logistics network does make initial database construction easier, and may also be useful when calibrating THUNDER output against mission level models.

The only graphical pre-processing tool (supplied with THUNDER) for constructing input files is **ttnet**, which uses the configuration file **ttnet.cfg** (usually located in the data directory). It allows an existing logistics network to be edited, or a new net to be constructed from scratch. The files worked on by **ttnet** are:

- **arcs.dat**: line segments joining nodes,
- **nodes.dat**: points on the net; may just be points where the direction of travel changes, or can be intersections etc,
- **trnsship.dat**: points where the method of transport changes, e.g., rail to sea; may have air defence sites, and
- **chokepts.dat**: points on the network which may be targetted for air attack; may have air defence sites.

Graphically, **ttnet** is very similar to **ttsm**, except that the ability to timestep through the results of a simulation has been replaced with the ability to edit the logistics network. Note that it displays information such as FLOT geometry and target status as it appears at the end of the simulation.

Figure 8 shows a **ttnet** window, taken from the Datasmall scenario. The "Edit an Arc" dialog box can be brought up by selecting the "Edit net" menu option, and then "Edit" from the "Arc" menu. One of the rail arcs has been selected for editing by following this procedure and then clicking on it. The dialog box then appears and the user can change any of the quantities which are in boxes with bold outlines. Note that the total arc length (labelled "Actual") is longer than the great circle arc between the two points (labelled "Line"). All other aspects of the network can be edited in a similar fashion.

In order to construct a new net, empty versions of these files are needed (i.e., all of the headings in the files are retained, but the actual data are removed). These can be simply constructed from example files (e.g., Data/Datasmall). Using **ttnet**, the network can be drawn on screen, hence allowing the setting of movement rates, temporary replacement bridges and so on. The data can then be saved to the relevant files in the correct input format for THUNDER.

Note that graphics transactions are needed in order for **ttnet** to run. This means that the scenario must be at a point where THUNDER can be executed. For this reason, it is not a good idea to start building a database from scratch, but from an existing scenario. A useful starting point is a totally "stripped down" scenario, which doesn't do anything useful (no missions fly, no logistics, no ground war etc).

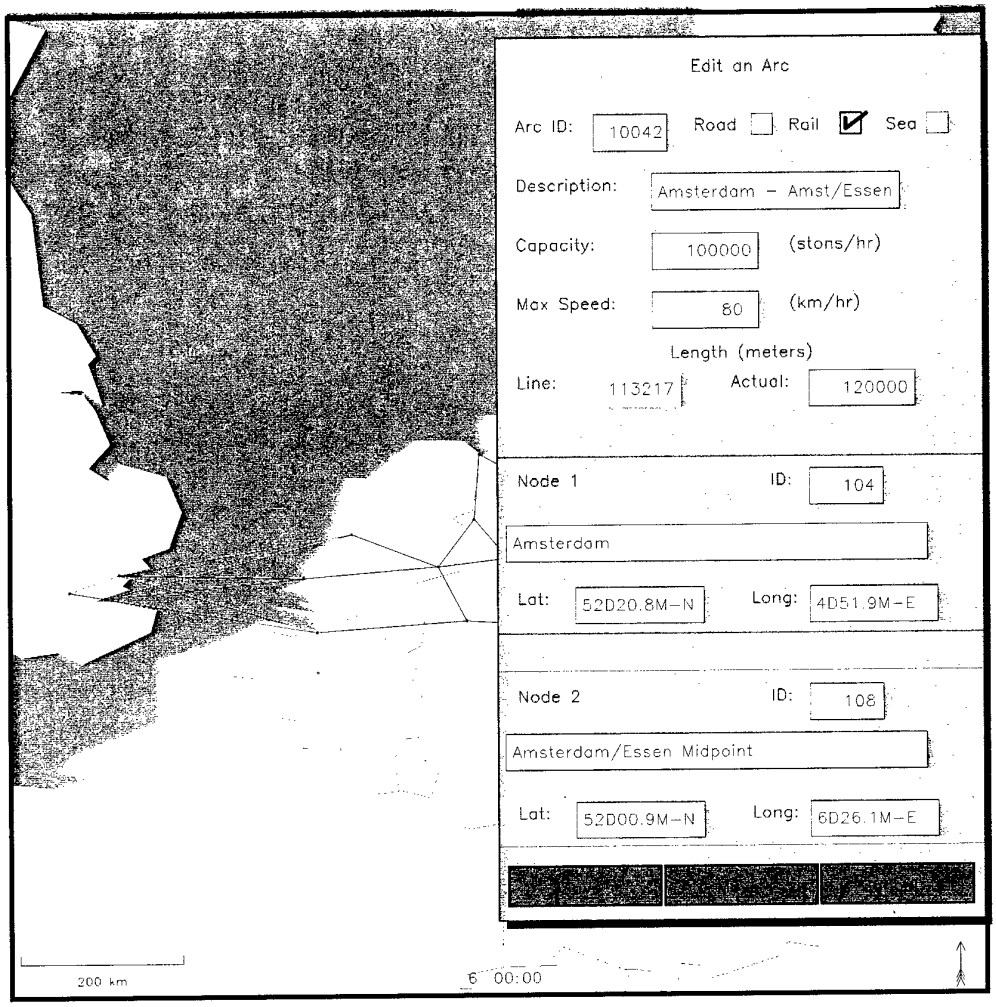

Figure 8: ttnet window demonstrating editing the network.

Another useful feature of **ttnet** is that individual points can be placed anywhere on the map. These points are saved to the file **points.dat**, and can be used to specify the location of various objects. Conversely, during construction of databases, **ttnet** may be used to show the location of objects such as orbit points for surveillance aircraft, and also used to construct the FLOT.

When setting up the logistics network, the analyst should not attempt to model every kink in the road, small bridge etc. THUNDER uses a form of shortest path algorithm to generate supply routes and so the addition of large numbers of unnecessary nodes can greatly increase runtime. Instead of drawing a road or rail arc as multiple arcs, the analyst can set the actual length of an arc to be larger than the straight line distance. If there is a series of chokepoints on a single road, it may be replaced by just one or two, as only one chokepoint really needs to be targetted and destroyed to break the road.

A logistics network for Australia has been constructed in THUNDER, and is fully described in reference [5].

Further files need to be modified in order to activate the logistics network. For example, ground combat and logistics rules are contained in the files **grdrules.dat** and **logrules.dat** (from version 6.5 onwards). Rules and planning files are discussed in Section 4.9.

4.4.1 Airlift

In addition to the land and sea based logistics network, airlift has been integrated into version 6.5 of the THUNDER model. In version 6.4, the air mission LIFT has been introduced, but no aircraft can have this as a real mission and no planning is done for it. Instead, fictitious air lift events can be set up in **lftevnts.dat**. No actual aircraft fly, and supplies just show up at predetermined locations at predetermined times. These supplies are also not drawn from the stocks contained in the logistics network.

Version 6.5 of THUNDER introduced real airlift missions. Supplies which are urgent, and cannot be delivered in time to their destination by the logistics network, are scheduled for airlift. The methodology used to prioritise supplies is quite complex, and is not discussed here.

Supplies can be airlifted to airbases, ground units and air defence complexes. Delivery methods include landing and unloading, and air drop. Ground units can also be lifted from one airbase to another. More capabilities are being added.

In order to determine which equipment can be lifted, every possible piece of cargo has an explicit weight and either an equipment size category (referring to the categories held in **equipsiz.dat**) or a size in loads spots (fractions of pallet). When an item is modelled as a critical resource, the pallet measure is used. Airlift capable platforms have both a maximum payload in short tons and a maximum load in load spots (i.e.,pallets). The analyst can use either weight or weight and volume for deciding airlift loads by setting the relevant flag in **logrules.dat**.

Not only must the airlifter be capable of lifting the cargo, but the departure airbase must have the capability to load and offload cargo. The relevant input data section for an example airbase is shown in Figure 9, together with an airlift relevant section of **logrules.dat**.

The four parameters in **airbase.dat** specific to airlift operations are:

- MOG: maximum number of lifters at the airbase at any one time; should consider other aircraft such as fighters which may be stationed there when setting this number,
- MHE.FACTOR: multiplying factor used to determine the time spent loading and unloading aircraft; higher factors result in shorter times,

- MAX.CARGO.BACKLOG: the maximum allowable backlog of cargo at the airbase, and
- LOAD.CLASSIFICATION.NUMBER: only lift aircraft with LCN less than this value can land at this airbase (specified in **typeac.dat**).

```
AIRBASE.DAT

ID.LIST...LATITUDE...LONGITUDE...AIR.CMD..FLYDIRECT..LOSABLE..NAME
   1001  24D42.6M-N  46D43.7M-E    1104       2         1    "RIYADH"

SIDE..MINE.EFF.CLASS..MIN.CEILING(M)..MIN.VIS(M)..CRATER.CREWS..T&L.ATTR.CLASS
   1       1901            300            100            10           10001
  AD.SITES..TYPE.ID...QTY
              :
              :
              :
  POL.TGTS......ID..NUM.ELTS..FRAC.CAP
               90001   13        1.0
  START.TIME.....MOG
      1.0      999
  START.TIME.....MHE.FACTOR
      1.0             1.0
  START.TIME.....MAX.CARGO.BACKLOG(STONS)
      1.0                 1000
  AIRBASE.LOAD.CLASSIFICATION.NUMBER    100
  END.AIRBASE



LOGRULES.DAT

CARGO.LOADING.FLAG..(1=WT.AND.VOLUME,2=WT.ONLY)    1
                    :
                    :
PREPLANNED.MOVEMENT.TO.ONLOAD.AB.FLAG(1=MAGIC,2=NETWORK):     1
```

Figure 9: Airlift relevant parameters in airbase.dat and logrules.dat.

Cargo is not drawn from the airbase stocks, but is supplied from logistics facilities. In order for cargo to get to an airbase for lift, the relevant logistics facilities must be explicitly linked to the airbase in the file **logrules.dat**. Supplies can move from the logistics facility to the airbase either by "magic" (i.e., they just appear) or using the road network if there is an arc between the logistics facility and a point at or near the airbase.

It is possible for an airlift mission to fly to multiple destinations, dropping off and picking up cargo along the way. THUNDER will attempt to fill any available space on an airlifter if there are supplies which need to be delivered. In order to fly to multiple points, the airlift platform must be made multiple target capable in **typeac.dat**, and the relevant multiple target parameters must be included in the aircraft planning factors

file **acplan.dat**. Two example multiple-stop airlift missions are shown in Figure 10. These were implemented as preplanned air tasking orders in the file **liftato.dat**.

The first example visits multiple airbases carrying pre-determined cargo, while the second simulates a "mail-run" (but may also pick up cargo along the way if it awaits airlift).

```
PREPLANNED.LIFT.ATOS.606
 SQDRN..SORTIES..T.O.T..ACTION....CARGO.TYPE./.ID..NUM..DEST.TY..DEST.ID..ZONE
 11024       6       2.8    M-AIRLAND  AMMO            0     10     AIRBASE     1001
                            M-AIRLAND  POL             0      5     AIRBASE     1002
                            M-DROP     AMMO            0      5     AIRBASE     1003
END.PREPLANNED.LIFT.ATOS


PREPLANNED.LIFT.ATOS.606
 SQDRN..SORTIES..T.O.T..ACTION....CARGO.TYPE./.ID..NUM..DEST.TY..DEST.ID..ZONE
 11024       1       3.0    M-POSITION NONE            0      0     AIRBASE     1001
                            M-OFFLOAD  ANY             0      0     AIRBASE     1002
                            M-OFFLOAD  ANY             0      0     AIRBASE     1003
                            M-OFFLOAD  ANY             0      0     AIRBASE     1001
END.PREPLANNED.LIFT.ATOS
```

Figure 10: Examples of multiple target airlift missions.

A simple airlift mission can be added to an existing database using the following method:

- define airlift capable platform in **typeac.dat,**
- edit all of the files needed when adding a new aircraft (Section 4.7),
- assign the airlift platform to a squadron in **squadron.dat,**
- edit the allocation of resources in **airalloc.dat** so that a command has responsibility for this squadron, and can fly lift missions,
- connect a logistics facility to the departure airbase, done by first defining a logistics facility in **logfac.dat** and then specifying the link in **logrules.dat, and**
- set up a preplanned ATO to shift some "stuff" in **liftato.dat.**

4.5 Fixed Targets

Fixed targets are created from the standard target instances defined in **stdtgt.dat** (Section 2). Files containing fixed targets are:

- **strat.dat:** strategic targets,
- **airbase.dat:** airbases,
- **trnsship.dat:** transshipment points, and
- **logfac.dat:** logistics facilities.

Air defence complexes also fit into the category of fixed targets, but are described seperately in Section 4.8.

Most fixed targets have a few basic elements in common:

- target repair functions (exponential, linear),
- target element separation functions,
- targets instances defined out of the standard target elements,
- air defence sites allocated to protect the target, and
- may be part of the integrated air defence network (IADS).

Each target defined in any one of the above files will need to contain all of this information.

Strategic targets are those which do not have an immediate impact on the war, but may have longer term consequences. As such, the destruction of a strategic target does not hamper the enemy's war effort in the short term. However, the whole objective of a BLUE campaign may be to destroy a particular RED strategic target. Such things as factories and oil fields may be defined as strategic targets. Note that there must be at least one strategic target per side, on the relevant sides of the FLOT. Even so, there is no requirement for them to be targetted during the air war.

Only the basic set of characteristics mentioned above are needed to set up strategic targets. More information must be provided for all other target types.

Airbases require all of the usual target information outlined above. They also need definitions of the elements which make them airbases:

- minimum ceiling and visibility for operations (weather can close an airbase),
- number of crater repair crews,
- takeoff and landing attrition class,
- repair resources on hand,
- runway lengths, widths, etc., and
- POL on hand.

Orientations of the runways are not required as THUNDER does not explicitly model aircraft taking off and landing.

Airbases also need to be allocated to an air command (the procedure for setting up commands is discussed in Section 4.1). They also require two flag entries for fly direct and losability. Note that if an airbase is overrun by the enemy, it cannot be used by either side and takes no further part in the simulation.

As well as the target information, logistics facilities require issue capacities, quantities of supplies on hand, and such things as issue capacity targets (such as cranes etc). Transshipment points require details concerning the transshipment process, such as the rate of transfer from one medium to the other (any of road, rail and ship).

Transshipment points and logistics facilities are not only fixed targets, but elements of the logistics network, which is discussed in Section 4.4.

4.6 Ground Objects

The first step in creating the ground objects is deciding on a command structure. This must be done simultaneously for ground and air commands, as they are all defined in the one file **command.dat**. Setting up the command structure breaks the battlefield into sectors, since each command has a nominal width on the battlefield, and there must be at least one on-line ground command on each side.

Much of the target allocation for the air war is done by command. It is therefore important to plan out the air campaign, in at least some detail, before setting up the command structure. For example, consider a RED ship which is being modelled as a strategic target, and is located in coastal waters.

A single BLUE air command may have both F-111s and P-3s allocated to it. If all other planning factors are not set up correctly, the F-111s may end up striking the ship, even though the BLUE "commander" desires only to use P-3s. (This is the sort of problem which can crop up as a result of THUNDER's automatic mission planning.)

One solution is to give the weapons carried by the F-111 a P_k of zero against the ship type target, even though this may not be strictly correct. This would mean that new P_k tables would need to be generated for a different scenario in which F-111s were supposed to strike ships. Also, the munition carried by both aircraft might be the same, and so duplicate munitions would be needed, further complicating the P_k tables.

A much better solution is to allocate the F-111s and P-3s to separate commands. Then, the P-3 command can be explicitly tasked to strike the ship, while the F-111 command is not. It is then a trivial exercise to modify the scenario so that the F-111s could strike the ship, and no dummy P_k tables have been introduced.

Once the command structure is set up, the next step is to define all of the ground units. This requires three basic files: **typeeq.dat**, **typeunit.dat**, and **units.dat**. In **typeeq.dat**,

the basic properties of each type of ground equipment (e.g., BLUE tank, RED fuel truck) is defined. Basic characteristics are such things as the category of equipment, repair functions, standard target IDs, size, weapons contained (e.g., 105mm tank gun), and a relative importance factor. The capabilities of all ground weapons are contained in the file **wpnvseq.dat**.

The characteristics of each unit type are defined in **typeunit.dat**. Here, the amount of equipment each unit type contains is defined, such as 10 tanks, 20 APCs etc. Also, air defence sites and C3 facilities are allocated to the units. Once all of the unit types have been defined, actual units are constructed from them and placed in **units.dat**. Each unit has a unit type, superior command, and an initial status and location. Units which have an on-line status are placed on the FLOT segment controlled by their superior command, and no location needs to be specified.

Although ground units can be made any size, the Attrition Calibration (ATCAL) tables used by THUNDER to calculate ground attrition are based on division sized units. It is thus not clear how they will fare with smaller units. This could be a particular problem when modelling ADF scenarios. It would be necessary to gain access to the classified ATCAL algorithms in order to make a fuller assessment of the problem.

It is possible to have ground unit types with no equipment and no icon (these icons are used by **ttsm** and **ttnet**). These "dummy" units can then be used to hold sections of the FLOT fixed throughout the simulation. See Section 5 for a description of using dummy units to maintain FLOT geometry.

Command, control and communication (C3) facilities are also defined during the building of ground objects (**typec3.dat**). Instances of these facilities can then be allocated to various targets, such as airbases, in the same manner as allocating air defence sites. These C3 facilities "process" information relevant to the ground war; each has a processing capacity in messages per hour. When C3 facilities are damaged, the downgraded message processing capacity can affect the ground war. The file **grdrules.dat** specifies this degradation for each side. See Section 4.9 for a further description of this file.

THUNDER can be run effectively with no ground war, either by giving the ground forces no equipment and no movement rates (all dummy units), or by setting the start of the ground war to a day after the scenario ends (e.g., day 99) in **grdrules.dat**. Even with no ground war, some ground objects are needed to allow the scenario to run. In particular, there must be at least one online unit and corresponding command per side, so that the battlefield width is finite.

4.7 Air Objects

Since THUNDER is primarily an air model, there is a lot of information which must be entered into the air objects files. Instead of going through each file in detail (and getting lost in these details), it is better to look at them from the point of view of actually adding new air objects. Several instances will be considered: adding a new aircraft, adding a new munition and adding a new air-based radar.

4.7.1 Adding a New Aircraft Type

Many files need to be modified in the THUNDER database in order to add a new aircraft type. Given that so many files need to be updated when adding an aircraft, it is a good idea to add several aircraft concurrently.

The procedure is:

- add basic aircraft properties to **typeac.dat**,
- add maintenance details to **acmaint.dat** and **acserv.dat**,
- add takeoff and landing P_k values in **tolandpk.dat**,
- if the aircraft has a new radar type, put this into **typerdr.dat** (Section 4.7.3),
- similarly, put any new jammers into **typejam.dat** and also add the new radar here, and
- if the aircraft has any new munition types, add these as indicated in Section 4.7.2.

The basic properties required for each aircraft are a mix of physical characteristics and basic doctrinal information. Physical characteristics include such things as radar and jammer type, radar cross section, take-off and landing requirements, and fuel loads and burn rates for various weapons loads. Doctrine influences the list of possible weapons loads and the set of delivery profiles.

When a new aircraft is added, THUNDER will not object automatically if **acserv.dat** is not updated, i.e., a service kit isn't specified for the new aircraft type. This will only become evident during a simulation. Only munitions which are in an aircraft service kit can be used during the simulation. For example, if an F/A-18 service kit contains no AIM-9Ms, and the only air-to-air configuration available for the aircraft is two AIM-9Ms, then the aircraft will not be able to fly any air-to-air missions (it will be left out of the planning process). It appears that squadrons deploy to airbases with one service kit.

There are several other related areas where some changes are needed. For example, P_k values for air-defence sites versus aircraft (class) need to be adjusted in **advsac.dat**. In order for this to happen, the aircraft needs to be given a target class. All aircraft target classes are set in **adclass.dat**. They are fully analyst definable and any number of aircraft classes can be specified. These classes are then used to determine the effects of air defences. Essentially, they break the aircraft into an air target element, just as a ground target is broken into standard target instances.

In order to get the new aircraft type to fly, modifications are necessary to several other files. This is particularly the case with high value asset (HVA) aircraft types, such as AEW&C and air-to-air refuellers. All of these issues are discussed in Section 5.

The basic file which must be updated in order to include the new aircraft type in the scenario is **squadron.dat**. This also allocates the squadron to an air command.

4.7.2 Adding a new munition

Adding munitions tasks can be broken into two categories:

1. adding a new load of existing munitions to an aircraft, and
2. adding a new load of new munitions to an aircraft.

The procedure in both cases is similar after the first step. Below is the procedure for adding the munitions. Note that when many of these files are adjusted, new entries are needed for the particular platform/weapon combination:

- if the munition is a new one, add to **airmunt.dat**,
- if the munition is of the air-to-ground type, adjust **airgrdpk.dat**, **tgtacq.dat** or **acqcurve.dat**, and **agwpnmin.dat**,
- if the munition is of the air-to-air type, adjust **airairpk.dat** and **rngadvn.dat**,
- anti-radiation munitions are treated differently to other air-to-ground types, and must be added separately to **harmpk.dat**, and
- mine munitions are also treated differently, and their characteristics must be added to **mine.dat** and **minepk.dat**.

All P_k files are for a given aircraft/munition pair. These numbers are usually taken from a combination of mission level models such as Suppressor and EADSIM [6,7,8] and JMEMs calculations for air-to-ground engagements [9].

4.7.3 Aircraft radars

Radar definitions are contained in the file **typerdr.dat**. This holds two different types of radars: aircraft based and ground based air defence (acquisition and fire control) radars. The file is broken into two main sections: the first contains basic detection properties for all radar types, while the second divides the radars into ground based (air-defence) and air based, and contains additional information necessary which defines the radar type.

Aircraft radars are only used against other aircraft, and all air-to-ground radars and ground based radars other that air defence ones are handled separately. Here, only air-to-air radars are considered; the details for air defence radars are discussed in Section 4.8.

The starting point for setting up all radars is to define a set of radar bands. These bands need not correspond to real radar bands and may include IR, UV and optical sensors (including the pilot's own eyes). It is up to the analyst to ensure that the radar bands

behave sensibly. Each aircraft must then be assigned a radar cross section (RCS) in **typeac.dat**, for each radar band.

By defining a visual band and a visual radar (i.e., eyes), aircraft which do not have any other air-to-air radars (possibly some ground attack aircraft and helicopters) can still locate other aircraft visually; jammer effectiveness against such a "radar" would need to be zero.

The procedure for adding a new aircraft radar is then:
- define its detection properties in **typerdr.dat**,
- set the jammer effectiveness versus this radar in **typejam.dat**, and
- if this radar operates in a new band, add the RCS for each aircraft type in **typeac.dat**.

As mentioned earlier, air-to-ground radar modes are not explicitly modelled as radars. Each target/aircraft pair has a maximum location distance (target location type, set in **typeac.dat**). A random draw is made using this and the actual distance to the target to see if the target is detected. Once detected, there is a more elaborate process to see if munitions can acquire the target. Data on target acquisition and detection are contained in **tgtacq.dat**. This models such things as FLIR pods etc. The target location, discrimination and acquisition process is discussed in some detail in Reference [1], Volume 2, chapter 5.

4.8 Air Defence Objects

Air defence objects can either be stand alone, or operate as a coherent whole when part of an integrated air defence system (IADS). Here, only the basic properties of air defence are discussed, and the subject of IADS is omitted.

All air defences are constructed from air defence site types. The file **typead.dat** defines these types, and contains the basic information such as which acquisition and fire control radars it needs, together with round properties such as number, range, weight, speed etc. Air defence radars are listed in **typerdr.dat**, which is discussed in Section 4.7.3.

The effectiveness of air defence sites versus aircraft classes is contained in the file **advsac.dat**. For HIGH resolution air defence, a P_k for each site versus each aircraft class by altitude band, weather band and day/night is needed. For LOW resolution, only a percentage kill is needed. As with the effects of air launched munitions, these P_k values need to be obtained from mission level models. The aircraft classes, along with air defence classes, are contained in the file **adclass.dat**.

THUNDER can handle incomplete data in the **advsac.dat** P_k tables. For each air defence site type, there is a simple YES/NO switch which can be used to turn on/off the P_k dependence on altitude, weather and/or day/night. Also, these P_k values are single

shot P_ks, and do not usually take into account multiple passes of an aircraft (THUNDER does not model such aircraft behaviour).

Most air defence sites are either allocated to protect specific targets, such as airbases, strategic targets, and transshipment points, or form part of a ground unit (mobile air defences). However, in THUNDER the analyst can specify a set of fixed air defence sites in the file **adcomplx.dat**. These sites cannot move, and can be used to defend geographic regions. In version 6.5, mobile air defence complexes have been introduced.

Somewhat non-intuitively, battlefield design has an impact on air defence engagements. In THUNDER, air defence sites are either terminal or area. During an air defence engagement, terminal sites are considered to be those associated with the target being attacked (e.g., the site allocated to a strategic target in **strat.dat**). Area air defence sites are all others in the target zone/sector, such as those associated with other targets, ground units, or separate air defence complexes (**adcomplx.dat**). An enemy flight group will have surface-to-air losses calculated separately for both types.

Prior to version 6.5, area air defence logic was based on the battlefield zone/sectors defined during the ground objects phase (strictly, when the command structure is set up). One consequence of this was that long range area air defence sites could not shoot outside their individual zone/sectors. If there are several long range air defence sites in a version 6.4 scenario, care should be taken to match to zone/sector shapes to the site envelopes (if possible).

Note that this zone/sector dependence is only a problem for area air defence engagements and does not affect the calculations of terminal defences when aircraft reach weapons delivery points. In version 6.5, the air defence logic has been modified to use the air network grid rather than zone/sectors.

Another interesting feature is that some parameters which impact on air defence calculations are contained in the ground rules file **grdrules.dat**.

In summary, the procedure for adding a new air defence site type, but not allocating it to any real targets, involves the following steps:

- define fire control and acquisition radars in **typerdr.dat**; it is possible to have a combined fire control and acquisition radar,
- add entries for **typejam.dat** corresponding to jammer effectiveness versus these radars,
- add P_k values for enemy anti-radiation missiles to **harmpk.dat,**
- build the air defence site in **typead.dat,**
- put P_k values for the site versus enemy aircraft in **advsac.dat,**
- add new air defence class to **adclass.dat** (if necessary), and
- add new engagement probabilities to **adaengpb.dat** (only needed if a new class has been defined).

At this stage, the new air defence site type should be sufficiently well defined to add it to any target. Further information is needed to model its contribution to an integrated air defence system.

Caution needs to be exercised in setting the range for air defence site radars. THUNDER calculates the line-of-sight for a radar based on the direct optical path. This does not take into account the ability of many radars to see further (by a factor of about $(4/3)^{1/2}$), and so may underestimate the radar detection ranges for surface-to-air radars.

4.8.1 Air Defence Calibration Mode

THUNDER possesses an air defence calibration mode which can be useful in verifying that air defence units are behaving as expected. This mode can be switched on by setting the ADF.CALIBRATION.MODE.ENABLE flag in the file **control.dat**. Figure 11 shows the relevant section of this file.

```
ADF.CALIBRATION.CONTROLS
    ADF.CALIBRATION.MODE.ENABLE                    NO
@     if enabled
@       only preplanned ATOs fly
@       all ADvsAC Pks = 0.0
@       unlimited AD ammo reloads
@       lethal SEAD disabled
@       air-to-ground disabled
@       special ADF calibration transactions enabled
@       ADF results averaged over multiple reps:
    REPLICATIONS.PER.ADF.CALCULATION                3
    USE.MANUAL.IADS.IADS.VALUES(YES,NO)           NO
    MANUAL.IADS.INTEGRATION.LEVEL(0.0-1.0)         0.7
    MANUAL.IADS.SECONDS.DELAY                     15.0
    MANUAL.IADS.INTIMIDATE                         0.0
  END.ADF.CALIBRATION.CONTROLS
```

Figure 11: Air defence calibration mode controls contained in the file control.dat.

The calibration mode does the following things:

- executes a user specified number of repetitions per air defence engagement calculation: This is performed during a single repetition of the THUNDER model,
- only preplanned ATOs fly (these need to be put into **ato.dat**),
- suppression of enemy air defence (SEAD), air-to-ground, and surface-to-air attrition is disabled, and
- unlimited air defence site ammunition reloads (up to the maximum amount stored for each transporter/erector/launcher, which is contained in **typead.dat**).

The calibration mode enables the analyst to fly a set number of aircraft through a particular region (straight line flight path) and assess the ability of air defences in that region to engage the aircraft. Note that, apart from the points made above, the full air

defence logic is used, including the effects of terrain masking. This enables the effects of intervisibility curves to be observed, through reduced fire control radar ranges.

Output is written to the transaction file **trans.trn** in the form of standard transactions. Some of these transactions are unique to calibration mode, while others refer to the general properties of the air defence engagement. The format of these analysis transactions is fully described in Reference [1], appendix G.

Some of the output from calibration mode includes:

- fire control radar detect ranges,
- first intercept range,
- number of engaged shooters, and
- number of shots per shooter.

For each of these numbers, a minimum, maximum and average value are returned. Figure 12 shows some sample transactions.

```
ADF810010 location stuff
  ADF810020 location stuff
  ADF810030 location stuff
  ADF810040  2001    2     2
  ADF810050    -1  7  5000  480  1  1001  7   1
  ADF810060 1.0000     0.
  ADF840010  2001  10000  10000  10000
  ADF840020  2001   8000   8000   8000
  ADF840030  2001     1.000      1      1
  ADF840040  2001  2.000  2.000  2.000
  ADF840050  2001     1.000     1.000     1.000
  ADF840060  2001     1.000     1.000     1.000
```

Figure 12: Sample air defence calibration transactions.

The exact format of the transactions, and their numerical codes, depends on the type of air defence engagement which has occurred (i.e., area or terminal). However, there are generally twelve transactions written for each engagement. The first three specify the location of the engagement, i.e., the air defence site. Transactions four to six contain details of the air defence site (type, integration level etc) and the target flight group (type, number, altitude, speed, etc). The last six transactions contain details of the engagements themselves, such as engagement ranges, number of shooters, etc.

4.8.2 Standoff weapons susceptibility to air defences

New to version 6.5 of THUNDER is the ability of standoff air-to-ground weapons (whilst in flight) to suffer attrition from air defences. This is handled quiet differently from aircraft attrition by air defences, and so is worthy of a brief description.

Various surface-to-air types are grouped into classes in **saclass.dat**. The file **muntsurv.dat** contains probabilities that air-to-ground munitions will suffer attrition due to these surface-to-air classes. These probabilities are not simple single shot P_{kS}, as in the case of **advsac.dat**, but are curves giving the probability of survival versus number of air defence TELs (of a particular type).

Note that munition flight is not modelled; when muntions are launched the number and type of air defence sites are used to calculate possible munition losses and all other munitions instantly enter the target attack logic. Munitions can only be "shot down" when launched from aircraft flying INT, OCA, STI, CAS and BAI missions.

4.9 Air and Ground Planning

Most of the "smarts" of the THUNDER input data are contained in a few planning files. These can be divided into ground and air planning files. Since the ground war is a simple deterministic piston model, ground planning is the easiest and is mostly contained in the file **grdrules.dat**, which was mentioned earlier. This file has a simple format which is easy for the analyst to modify.

Since THUNDER does all of its own air mission planning, a great deal of information is needed for the air planning process. The model relies heavily on analyst input in deciding how mission planning is to proceed. It is beyond the scope of this work to go into the air mission planning files in detail, hence here some key points are noted.

In **squadron.dat** a number of mission classes are defined. Squadrons are then allocated to only one mission class. A combination of this class and aircraft effectiveness data from **squadron.dat** and **typeac.dat** determine the missions that the aircraft can actually fly (together with munition resource at the squadron's airbase).

The file **airalloc.dat** specifies the allocation of mission class resources, for each command, versus time. For example, for the first few days of a conflict, the mission class MULTI.ROLE might be allocated 100% to offensive counter air (OCA). Once the enemy no longer poses an air threat, the allocation may be changed to 20% OCA, 80% strategic target interdiction (STI). The format of this file is easy to grasp, and it is probably the simplest of the air planning files.

In version 6.4, allocation can only be varied by time. From version 6.5 onwards, it is be possible to vary air allocation based on the achievement of certain objectives, such as the suppression of enemy air defences or achieving air superiority. This has necessitated the modification of many of the other air planning files. In most of these files the only change is replacing the TIME instruction with TIME.OR.CONDITION so that the allocation of resources can be varied using either method. Modified files include **acplan.dat**, **airalloc.dat**, **airplan.dat** and **cbg.dat**. The conditions which may affect air planning are contained in the new file **aprules.dat**.

The use of conditional logic for the allocation of air resources offers a significant improvement. Previously, the analyst had to run THUNDER, see when the condition changed, modify the planning files manually by changing the allocations versus time, and then re-run the model to check. This was a tedious and non-optimal process; the timings could be slightly out and, since THUNDER is a stochastic model, the correct times for one repetition of the model may not be the correct times for many repetitions.

Rule based air planning also allows the analyst to extend THUNDER beyond its present capabilities through some creative modelling. For example, THUNDER does not model ground control intercept (GCI) radars as explicit radar installations. This means that these radars cannot be targetted for attack, and thus the enemy's early warning capability cannot be degraded. Using rule based air planning, it is possible to set up a target associated with the GCI radar and, if this is destroyed by BLUE, change the allocation of RED air resources so that their ability to fly long-range intercept missions is degraded.

Two files of great importance (and with similar names) are **airplan.dat** and **acplan.dat**. The first of these, **airplan.dat**, contains mission-specific parameters for each command. Such things as the mix of target elements to attack in OCA missions are contained here. The second file, **acplan.dat**, contains planning factors for individual aircraft types. Not all aircraft types require their own specific planning factors, but they are required for all high value assets (HVAs) such as AEW&C platforms and tankers.

It is possible to override the automatic mission planning sequence using preplanned air tasking orders (ATOs). These are contained in the file **ato.dat**, and specify which squadron will carry out the ATO, and such parameters as the mission, target and target element etc. Note that preplanned ATOs need to be entered in advance and so may not really fit into the rest of the air campaign. Also, factors such as aircraft range may be overlooked. Carefully constructed preplanned ATOs can be of use for performing unusual missions or during database construction and testing.

Airlift missions (in THUNDER version 6.5 and later) have their own prelanned ATO file, called **liftato.dat**.

See the sections on ground war planning (Reference 1, section 2.6, page 204) and air war planning (Reference 1, volume 2, section 2.7, pages 204-205) for a complete list of files involved in planning.

4.10 Debugging Scenarios

Errors in the input data are indicated in two ways during input file reading by THUNDER. First, apparent inconsistencies in the data, e.g., nonexistent aircraft types, disconnected logistics network elements etc, are indicated in the file **debug.out**. This file contains the location of the error (filename and line) and so can be very useful

when debugging an input database. Note that some of these discrepancies may have been deliberate on the part of the user, and so are not errors as such. All of the supplied examples contain many such discrepancies (run THUNDER using one of the supplied databases and view **debug.out**).

Note that errors written to debug.out can be indicated in two different ways. If the error is serious, THUNDER may quit during execution, with a message of the form

```
THUNDER: ABNORMAL TERMINATION - check debug.out file
```

or, the model may run without incident, but some errors are still written to the file.

Sometimes, there may be an error which causes a crash but which does not appear in **debug.out**. If this is the case, the second error file, **game.err,** should be examined. It indicates which routine and line number was being executed when the problem occurred, and so gives some idea of the error type. This may not be an input routine if there is a subtle bug in the data. Unlike **debug.out**, **game.err** is only written if THUNDER actually terminates abruptly. The following message will be displayed:

```
THUNDER: PROGRAMMED ABORT - check game.err file
```

Occasionally, when this error message is displayed it is actually more helpful to check the debug.out file.

In summary, there are several steps to follow when debugging a new or modified database:

- turn on data echo reports in **control.dat** to ensure that THUNDER is reading in the data correctly; execute **ttrun** and check,
- if THUNDER actually crashes, check **debug.out** and/or **game.err**, and
- if THUNDER does not crash, but the output does not appear to be correct, check **debug.out.**

If THUNDER does crash while reading input data, it may not crash at the exact point where the error occurs. Since the input data is divided into a large number of files, an input error may only occur when data is read in which is inconsistent with previously read data. It is thus useful to know what the sequence is for THUNDER file input. The sequence is shown in Figure 13.

1 lftevnts.dat	22 urgcurve.dat	43 bpi.dat	64 acqcurve.dat
2 isrevnts.dat	23 typec3.dat	44 airbase.dat	65 agwpnmin.dat
3 control.dat	24 command.dat	45 acserv.dat	66 logrules.dat
4 seedval.dat	25 train.dat	46 squadron.dat	67 airalloc.dat
5 stdtgt.dat	26 logfac.dat	47 muntsurv.dat	68 ocatgts.dat
6 typerdr.dat	27 adcomplx.dat	48 aprules.dat	69 stitgts.dat
7 typejam.dat	28 strat.dat	49 cbg.dat	70 zsprior.dat
8 adclass.dat	29 weather.dat	50 adsector.dat	71 intdepth.dat
9 typead.dat	30 typeeq.dat	51 saclass.dat	72 airplan.dat
10 adaengpb.dat	31 typeunit.dat	52 critres.dat	73 tgtavail.dat
11 battlfld.dat	32 equipsiz.dat	53 airrules.dat	74 acplan.dat
12 gridcap.dat	33 wpnvseq.dat	54 airairpk.dat	75 srec.dat
13 mobility.dat	34 units.dat	55 rngadvn.dat	76 airnet.dat
14 intcurve.dat	35 grdrules.dat	56 detect.dat	77 ato.dat
15 intervis.dat	36 percept.dat	57 advsac.dat	78 liftato.dat
16 density.dat	37 isrsens.dat	58 airgrdpk.dat	79 isrnodes.dat
17 nodes.dat	38 airmunt.dat	59 harmpk.dat	80 isrtgts.dat
18 trnsship.dat	39 typeac.dat	60 minepk.dat	81 satellit.dat
19 arcs.dat	40 acmaint.dat	61 tolandpk.dat	82 isreff.dat
20 ckpts.dat	41 tbmunit.dat	62 wmd.dat	83 isrplan.dat
21 mine.dat	42 tbmdet.dat	63 tgtacq.dat	84 isrinit.dat

Figure 13: File input order during THUNDER execution.

THUNDER reads all of the input files in the sequence indicated above. If there are no syntax errors in a particular file, and it does not contain information which is inconsistent with previous files, then it is accepted. Problems start to occur when inconsistencies are detected. For example, **typead.dat** (file 9) may reference a radar type not contained in **typerdr.dat** (file 6) and an error message will be generated. Checking **debug.out** in such cases will quickly reveal the source of the error.

Occasionally, THUNDER will crash when reading in the first few lines of a file, where there is no actual data (just headers). This is usually due to an error in the previous file on the list, since THUNDER concatenates all of the files into **input.dat** before reading them in.

5. Limitations

At the highest level, a campaign model such as THUNDER attempts to model the entirety of a conflict, and so cannot get "bogged down" in the details. This means that it is sometimes difficult to represent accurately particular systems and patterns of behaviour.

Many of THUNDER's limitations stem from the historical origins of the model, since it was developed as a campaign model for the USAF. Also, the model's initial development during the cold war era has influenced the types of conflicts envisaged as needing campaign analysis and hence the algorithms implemented.

This section describes some of the limitations of the THUNDER model. While not exhaustive, the list (together with Section 4) should give a feeling for the types of problems which can and cannot be handled using THUNDER.

Although also a weakness, one of the strengths of the model is that it does not concern itself with a high level of detail. In effect, many things in THUNDER are modelled for effect rather than accuracy. This does leave the way open for the analyst to emulate the effects of systems not explicitly modelled by THUNDER.

The solutions to such problems are almost always highly scenario dependent. Here, the solutions to two such problems encountered while performing operational studies [11] are described in some detail; modelling individual Ground Control Intercept (GCI) radars, and modelling large surface combatants. These solutions are intended to give a feel for the types of manipulations possible using THUNDER. In particular, the GCI discussion illustrates the utility of THUNDER's inbuilt scripting language.

5.1 Ground War

THUNDER was originally produced in order to analyse traditional NATO/Warsaw Pact conflicts during the cold war era. As such, the ground war model was developed to represent attrition-type warfare between large formations of troops and armoured vehicles across a limited front. Based on the US Army Atrittion Calibration (ATCAL) methodology, the model is largely a set of heterogeneous Lanchester equations [10]. The two main limitations of the ground war in THUNDER resulting from this focus are:

- lack of manoeuvre warfare, and
- calibrated for large scale forces only.

The lack of manoeuvre warfare in THUNDER is an artefact of the linear piston model used. This deficiency is also evident in the inability to model amphibious and airborne operations, and any movement of troops behind enemy lines. Also, the focus on attrition algorithms is appropriate for large-scale conflict, but may not be so for smaller scale operations.

There are further problems associated with interactions between the ground battlefield and the air war. This is particularly a problem when using versions older than 6.5. Specifically, the zone sector arrangement on the battlefield can have an impact on the allowed surface-to-air engagements as air defence sites cannot shoot out of their zone sector in version 6.4. This is particularly a problem in modelling modern long-ranged

systems such as Patriot. Also, all air defences not allocated to the terminal targets are allocated to zone sector defence if the air defences are fully integrated.

Consider a specific example of two RED targets, seperated by approximately 100km. Target 1 is to be attacked and has no air defences, while target 2 is not attacked but has an air defence site allocated to it (with a terminal radius << 100km). Both of these targets are located within the same battlefield zone sector and all air defences are integrated (IADS). Upon execution it is observed that the BLUE strike package attacks target 1 and suffers surface-to-air attrition.

The explanation is that RED air defence assets not connected with the target are allocated to the zone sector for AREA air defence. Thus the BLUE flight group suffers AREA attrition from these when it enters the zone sector and the air defence lethal radius. This radius is not centred on target 2 but is just located in the zone sector.

Such a problem is possibly not an issue for many US scenarios with small zone sectors. A possible solution is to divide zone sectors so that targets are located either individually in them or zone sector sizes are comparable to air defence site lethal radii.

5.2 Air War

Since THUNDER was originally developed for the USAF to model air campaigns, it is not surprising that the air war algorithms are relatively extensive and that numerous systems and capabilities can be represented. The large number of mission types, together with very flexible command structures and resource allocation rules allow many varied missions to be flown.

There are still some deficiencies, though. These include such things as:

- a lack of third party targetting,
- a lack of dynamic updates to aircraft already enroute,
- the relative altitude of aircraft is not considered in air-to-air engagements; in effect, all aircraft have a full look down/shoot down capability),
- ground control intercept radars cannot be targetted or destroyed by air attacks; thus it is not possible to completely disable an early warning system, and
- ground based radars are all assumed to be at sea-level, limiting their radar horizon; the horizon is further limited to direct line-of-sight.

At the heart of THUNDER is an automatic Air Tasking Order (ATO) generator which allocates air sorties against targets based on analyst input targeting priorities and mission allocations. The algorithms used are designed to allocate optimally sorties to the perceived target set, and support assets (refuelling, escorts, jammers etc) are then allocated to these primary packages.

Such a focus on an optimal allocation of assets may be well-suited to large-scale air campaigns but may not necessarily be appropriate to smaller operations. Smaller-scale conflicts where there is a scarcity of air assets may require methods which pay more attention to risks in calculating expected payoffs. Using the current methodology and applying the model there is little optimisation by THUNDER itself as the analyst needs to set target priorities in such as way that the allocation of sorties is correct. In effect, the analyst is determining directly the sorties flown.

5.3 Naval Aspects

Naval warfare is THUNDER's weakest area. In version 6.5, only carrier battle groups (CBGs) and supply ships are explicitly modelled. Even these entities have a number of limitations:

- If a ship is to be targetted for air attack, particular care needs to be taken when setting up its target structure (since they are treated just like land targets); acquisition and discrimination curves should also be adjusted to take account of the particular marine environment,
- Much of the movement of carrier battle groups needs to be hard coded into the database in the form of scripts; to some extent movement can be made dependent on developments during the campaign, but this is currently limited,
- Supply ships cannot be escorted by any other vessels. It is, however, possible to have a ground unit travel by supply ship and this unit may have air defence assets.

Carrier battle groups are modelled as moving airbases, with the appropriate target structure etc. Using this as a basis, it may be possible to construct other large surface combatants. Even so, it would be overextending the model to try and simulate any purely naval activities. Ships of this kind should only be modelled if required to exert an influence on an extended air campaign.

5.4 Modelling GCI Radars

THUNDER models Ground Controlled Intercept (GCI) radars differently to airborne and dedicated air defence radar types (they are not contained in typerdr.dat). GCI radar range is not computed as a radius around the radar, but a distance from the FLOT (stored in airrules.dat) and the analyst must specify the high and low altitude ranges correctly as GCI radars do not use the standard THUNDER radar horizon calculation. This limited model of GCI radars is yet another artefact of THUNDER's origins as a cold war analysis tool.

There are two main problems associated with the conventional THUNDER model for GCI radars:

- radars are not discrete and so may not possess realistic coverage patterns, and
- radars are not interdictable.

There is no unique or universal solution to this problem. However, certain constrained solutions have been possible for individual study scenarios.

The main problem encountered when attempting to interdict GCI radars is that they are not points on the map which can be targetted as there are no actual targets allocated to them. Also, GCI radar parameters (specified in airrules.dat) cannot be made time or condition dependent. Luckily, many THUNDER input files make use of a simple but powerful scripting language. This means that scenario elements such as strategic targets can have an influence through the conditional logic introduced by scripted input.

The GCI modelling approach suggested here is to effectively associate isolated strategic targets with GCI radars and to use the scripted THUNDER language to influence the air war accordingly; when these targets are damaged or destroyed the ability of the enemy to launch interceptors is reduced. Note that GCI radars can also affect surface-to-air engagements if they are part of an integrated air defence system (IADS). It is also possible to model the reduction of IADS effectiveness when GCI radars are destroyed using the same types of rules.

The remainder of this section describes the solution in some detail. The three steps involved are:

- define the relevant stragtegic target in **strat.dat,**
- set up the appropriate air planning rules in **aprules.dat** so that desctruction of the target has the correct influence on the air war, and
- add an air allocation to **airalloc.dat** indicating what action should be taken if the rule becomes TRUE.

Setting up the strategic target is trivial and is not discussed here. For the purpose of the discussion, assume that target 21111 is the GCI radar defined in **strat.dat**. The GCI radar is owned by RED and so is subject to BLUE interdiction. Also, assume that the relevant air-to-ground P_k data has been added to **airgrdpk.dat**. The next step is to add a rule to **aprules.dat** governing the destruction of this radar. Figure 14 shows the relevant excerpt from the file.

Once the new rule has been added it can be used in **airalloc.dat** to influence the allocation of air assets. Figure 15 illustrates a simple allocation change based on the rule. Before GCI destruction, RED has allocated air superiority assets to 50% DCA (strip alert) and 50% BARCAP missions. If the GCI target is destroyed then RED is forced to allocate 100% of assets to BARCAP as no detections can be made in time to send up DCA missions. Note that if RED possesses other early warning assets such as airborne AEW&C it is possible to add further subrules.

```
    :
    :
GROUP.NAME     RED.GCI.TGTS
GROUP.TYPE     STRAT.TGT
LIST.OF.IDS
    21111
END.GROUP
    :
    :
    :
BEGIN.RED.AIR.PLANNING.RULES
    BEGIN.OBJECT.VARIABLES
        RED.GCI.TGTS.DRAWDOWN
    END.OBJECT.VARIABLES

    RULE.NAME   RED.GCI.TGTS.DRAWDOWN
    BEGIN.RULE
        IF  STRAT.TGT.COUNT(RED.GCI.TGTS) = 0 THEN
            LET RED.GCI.TGTS.DRAWDOWN = TRUE
        ELSE
            LET RED.GCI.TGTS.DRAWDOWN = FALSE
        ENDIF
        EXIT
    END.RULE
END.RED.AIR.PLANNING.RULES
```

Figure 14: Execerpt from aprules.dat showi.ng the logic needed for a GCI destruction rule.

```
BEGIN.AIR.PLAN
    AIR.COMMAND  2100              @ relevant RED air command
    GROUND.COMMANDS  2001 2002     @ supported ground commands

@ this plan sets up the initial allocation of resources
    BEGIN.AIR.PLAN.CONDITION
        TIME.OR.CONDITION.NAME  1.000
        AIR.SUPERIORITY
            50  DCA
            50  BARCAP
        END.MISSION.CLASS
          :
          :
    END.AIR.PLAN.CONDITION

@ this plan changes the allocation if RED.GCI.TGTS.DRAWDOWN becomes TRUE
    BEGIN.AIR.PLAN.CONDITION
        TIME.OR.CONDITION.NAME  RED.GCI.TGTS.DRAWDOWN
        AIR.SUPERIORITY
           100  BARCAP
        END.MISSION.CLASS
          :
          :
    END.AIR.PLAN.CONDITION
END.AIR.PLAN
```

Figure 15: GCI destruction impacts on air allocation.

If the GCI radar is also part of the IADS, i.e. it provides information used in surface-to-air engagements or acts as part of the overall command system, then the impact of target destruction on the IADS should also be included. The approach used is very similar to that employed above. The only extra complication is that the relevant strategic target needs an IADS number in **strat.dat**. The solution is not discussed here as a very similar approach is used to simulate air defence ships and is considered in Section 5.5.

5.5 Modelling Large Surface Combatants

Naval warfare is not really modelled in Thunder and so a unique solution applicable to all scenarios is not possible. Here, a particular scenario is considered where the surface combatants are largely functioning as "picket ships", i.e. they engage in surface-to-air combat and also may be targetted by air strike missions. The ships are assumed to be protecting land targets, and so patrol relativly well-defined areas.

THUNDER does possess a capability for modelling Carrier Battle Broups (CBGs). These are treated as moving airbases, which may possess their own air defences and aircraft. There are problems associated with this model, such as the need to carefully script movement of the CBG, the fact that they cannot be sunk, and that they cannot be located using maritime patrol type missions. These problems mean that the CBG model may be a poor representation of a true CBG. As such they are a very poor representation of a smaller surface combatant.

A different approach has been used in studies where the scenarios are similar to that proposed above. Essentially, the approach is to approximate the impact that the ship has on the air war. The relevant properties of the ship are:

- mobile SAM with associated radars,
- moves within a relatively well-defined patrol area, and
- exact location (aviodance and targeting) is unknown.

In THUNDER these effects are captured by representing the ship as a strategic target at a fixed location within the scenario patrol area. The ship possesses a long-ranged SAM covering the sector, but with a much reduced engagement probability representing the fact that the ship could be randomly distributed within the patrol region. SAM range and engagement probability can be chosen to represent the true system range and probability given the sector size etc. Also, the ability of the strike aircraft to locate and acquire the target is modified to represent some uncertaintly with respect to the current ship location.

The ship can thus be constructed by defining the appropriately modified naval SAM unit in **typead.dat**, constructing a ship strategic target in **strat.dat**, and then linking the two through an integrated air defence sector in **adsector.dat**. This ensures that destruction of the strategic target disables the SAM. The actual SAM unit itself is not

subjected to attack. The most difficult step is constructing the rule in **adsector.dat**, and so an example is provided in Figure 16.

```
AIR.DEFENSE.SECTORS.105

   NUMBER.OF.AIR.DEFENSE.SECTORS:     1
   SECTOR.ID..SECTOR.NAME..POINT.LAT....POINT.LONG.
      25001    "RED SHIP"    (LIST OF POINTS)
            END.SECTOR.POINTS
   END.AIR.DEFENSE.SECTORS

   NUMBER.OF.IADS.OBJECTS:      4

      25101 RED_NAVAL_TEL          GROUND
      25102 RED_NAVAL_ACQ_RDR      GROUND
      25103 RED_NAVAL_FC_RDR       GROUND
      26001 RED_SHIP_TGT           GROUND

   END.IADS.OBJECTS

   IADS.RULE.SEGMENTS
      BEGIN.SECTORS
         25001              @ RED SHIP SECTOR
         BEGIN.SECTOR.IADS.RULES.(DEFAULT.ACTION.IS.INTEGRATED=1.0)
            DEFINE Ship
            LET Ship = COUNT(RED_SHIP_TGT)
            IF Ship = 0 THEN            @ SHIP HAS BEEN DESTROYED
               LET INTEGRATION = 0
            ENDIF
            EXIT
         END.SECTOR.IADS.RULES
        :
        :
   END.RULE.SEGMENTS

        :
        :

END.AIR.DEFENSE.SECTOR.FILE
```

Figure 16: Air defence sector rules for simulating a picket ship.

6. Discussion/Conclusions

THUNDER is a very capable campaign level model, incorporating air and land forces, a joint command structure and an extensive logistics network. The model is well supported, and comes complete with extensive documentation.

The model is supplied with a good set of both text and graphics based post-processing tools. However, there is little in the way of pre-processing tools to speed up database construction. Such tools were not deemed necessary by the developers as most US database construction is done by a dedicated group of analysts with high levels of model familiarity.

Since so many aspects of a campaign are modelled by THUNDER, there is a correspondingly large requirement for input data, even if they are at low resolution. It is a major exercise to set up a new database from the beginning.

Fortunately, many elements of the database are contained in library files from which elements may be selected for any given scenario. Once these files are constructed for one scenario, it is a relatively quick task to change scenarios by adding or removing only a few elements.

The documentation supplied with the model makes database construction a well-defined, if somewhat daunting, problem. Additional detail contained in this report provides further information to assist the analyst with this process. Much of the discussion is centred around adding a single new element to an existing database.

Even though THUNDER is a very capable model, it cannot do everything. Due mostly to THUNDER's origins in the USAF, the air war is understandably the best modelled part of the campaign. Logistics is equally well modelled, and is undergoing rapid improvements. The ground war is modelled adequately for large scale conflicts, but may not be suitable for smaller scenarios, and naval warfare modelling is very restricted, almost non-existent. Since it is a high level model, essentially "modelling for effect", it may be possible to overcome many of these perceived weaknesses.

7. References

1. *THUNDER Analyst Manual*, Version 6.5, System Simulation Solutions Incorporated, Washington, 1997.

2. Joint Pub 3-56.1: *Command and Control for Joint Air Operations*, United States Joint Chiefs of Staff, 14 November 1994.

3. Griggs, B. J., Parnell, G. S., Lehmkuhl, L. J., (1997). *An Air Mission Planning Algorithm using Decision Analysis and Mixed Integer Programming*, Operations Research, Vol. 45, No. 5, pp. 662-676.

4. Maguire, P., *THUNDER Utilities*, DSTO GD (in preparation).

5. Perry, N., Bocquet, S., *Modelling of Australian Infrastructure in the THUNDER Air Campaign Simulation*, DSTO TR (in preparation).

6. Whitehurst, R., Phipps, J., Kowalenko, V., *A Guide to Using Suppressor in Studies of Large Scale Air Operations*, DSTO-GD-0129.

7. *EADSIM: Extended Air Defense Simulation Methodology Manual,* Version 5.00, Teledyne Brown Engineering Defense Programs, November 1995.

8. Blanchonette, P., Whitehurst, R., *A Comparison of the Features of Extended Air Defence Simulation (EADSIM) and Suppressor: A Guide for DSTO,* DSTO-TR-0533.

9. *Joint Munitions Effectiveness Manual, Air-to-Surface,* JMEM/AS Weaponeering System (JAWS), CD-ROM Version 1.0, 1 November 1995 (SECRET).

10. Taylor, J. G. (1983). *Lanchester Models of Warfare.* Operations Research Society of America. Arlington, Virginia

11. Maguire P., Bocquet S., Perry N., *A Campaign Level Analysis of AGM-142 Stock Requirements for the ADF,* Client Report AOD 98/12 (SECRET AUSTEO)

DISTRIBUTION LIST

Using THUNDER for Campaign Studies

Peter Maguire

AUSTRALIA

DEFENCE ORGANISATION

Task Sponsor

Director General Aerospace Development

S&T Program

Chief Defence Scientist
FAS Science Policy
AS Science Corporate Management
} shared copy

Director General Science Policy Development
Counsellor Defence Science, London (Doc Data Sheet)
Counsellor Defence Science, Washington (Doc Data Sheet)
Scientific Adviser to MRDC Thailand (Doc Data Sheet)
Scientific Adviser Policy and Command
Navy Scientific Adviser (Doc Data Sheet and distribution list only)
Scientific Adviser - Army (Doc Data Sheet and distribution list only)
Air Force Scientific Adviser
Director Trials

Aeronautical and Maritime Research Laboratory
Director

Chief of Air Operations Division
Research Leader Air Operations Analysis
Head Mission and Campaign Analysis
Task Manager, L. Halprin (AOD)
P. Maguire

S. Bocquet
N. Perry

DSTO Library and Archives
Library Fishermans Bend (Doc Data Sheet)
Library Maribyrnong (Doc Data Sheet)
Library Salisbury (1 copy)
Australian Archives
Library, MOD, Pyrmont (Doc Data Sheet only)
US Defense Technical Information Center (2 copies)
UK Defence Research Information Centre, (2 copies)
Canada Defence Scientific Information Service, (1 copy)
NZ Defence Information Centre, (1 copy)
National Library of Australia, (Doc Data Sheet)

Capability Systems Staff
Director General Maritime Development (Doc Data Sheet only)
Director General C3I Development (Doc Data Sheet only)

Army
ASNSO ABCA, Puckapunyal, (4 copies)
SO (Science), DJFHQ(L), MILPO Enoggera, Queensland 4051 (Doc Data Sheet only)

Intelligence Program
DGSTA Defence Intelligence Organisation
Manager, Information Centre, Defence Intelligence Organisation

Corporate Support Program
Library Manager, DLS-Canberra

UNIVERSITIES AND COLLEGES
Australian Defence Force Academy
Library
Head of Aerospace and Mechanical Engineering
Hargrave Library, Monash University (Doc Data Sheet only)
Librarian, Flinders University

OTHER ORGANISATIONS
NASA (Canberra)
AusInfo

OUTSIDE AUSTRALIA

ABSTRACTING AND INFORMATION ORGANISATIONS
Library, Chemical Abstracts Reference Service
Engineering Societies Library, US
Materials Information, Cambridge Scientific Abstracts, US
Documents Librarian, The Center for Research Libraries, US

INFORMATION EXCHANGE AGREEMENT PARTNERS
Acquisitions Unit, Science Reference and Information Service, UK
Library - Exchange Desk, National Institute of Standards and Technology, US

SPARES (5 copies)

Total number of copies: 45

DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION
DOCUMENT CONTROL DATA

1. PRIVACY MARKING/CAVEAT (OF DOCUMENT)

2. TITLE

Using THUNDER for Campaign Studies

3. SECURITY CLASSIFICATION (FOR UNCLASSIFIED REPORTS THAT ARE LIMITED RELEASE USE (L) NEXT TO DOCUMENT CLASSIFICATION)

Document	(U)
Title	(U)
Abstract	(U)

4. AUTHOR(S)

Peter Maguire

5. CORPORATE AUTHOR

Aeronautical and Maritime Research Laboratory
PO Box 4331
Melbourne Vic 3001 Australia

6a. DSTO NUMBER	6b. AR NUMBER	6c. TYPE OF REPORT	7. DOCUMENT DATE
DSTO-TN-0303	AR-011-553	Technical Note	August 2000

8. FILE NUMBER	9. TASK NUMBER	10. TASK SPONSOR	11. NO. OF PAGES	12. NO. OF REFERENCES
M1/9/501	AIR 96/187	DGAD	57	11

13. URL on the World Wide Web

http://www.dsto.defence.gov.au/corporate/reports/DSTO-TN-0303.pdf

14. RELEASE AUTHORITY

Chief, Air Operations Division

15. SECONDARY RELEASE STATEMENT OF THIS DOCUMENT

Approved for public release

OVERSEAS ENQUIRIES OUTSIDE STATED LIMITATIONS SHOULD BE REFERRED THROUGH DOCUMENT EXCHANGE, PO BOX 1500, SALISBURY, SA 5108

16. DELIBERATE ANNOUNCEMENT

No Limitations

17. CASUAL ANNOUNCEMENT Yes

18. DEFTEST DESCRIPTORS

models
campaign analysis
scenarios

19. ABSTRACT

This report provides a detailed guide to the application of the USAF model, THUNDER, to air campaign analysis. All of the steps required to perform an analysis are considered, from setting up and debugging the extensive databases through to running the model and its associated post-processors.

Some of the model's fundamental limitations are also outlined. These limitations place important constraints on the types of campaign analyses which can be conducted with the model. While it is not possible to provide comprehensive solutions to all of these limitations, the flexibility of the model often allows scenario-specific solutions to be implemented. As an illustration, two such problems and their solutions are described in detail.